Tampere University

Ainaz Ebrahimi

# DYNAMIC USER PREFERENCES OPTIMIZATION IN TIME-AWARE RECOMMENDATION SYSTEMS

Faculty of Information Technology and
Communications Sciences (ITC)
Master's Thesis
Supervisor: Prof. Konstantinos Stefanidis
October 2024

# ABSTRACT

Ainaz Ebrahimi: **Dynamic User Preferences Optimization in Time-Aware Recommendation Systems**
Master's thesis
Tampere University
Master's Degree Programme in Data Science
October 2024

The integration of recommendation technology within Internet of Things (IoT) environments has been significantly accelerated by advancements in network technology. These recommendation systems, which predict user preferences to deliver personalized suggestions, play a critical role in mitigating information overload by presenting the most relevant products or services to users. Traditional recommendation algorithms—including collaborative filtering, content-based filtering, and knowledge-based approaches—have proven effective but face challenges related to data sparsity, scalability, and the evolving nature of user preferences.

This research addresses these limitations by enhancing Time Correlation Coefficient (TCC) models to better capture temporal dynamics. Unlike existing models that rely on a single attenuation coefficient, this study introduces customized coefficients tailored to each user, allowing the system to more precisely capture individual preference shifts over time. Furthermore, by integrating the TCC with item similarity scores, the approach aims to enhance both the relevance and accuracy of the recommendations. Four innovative methodologies are proposed to refine the TCC framework: dynamic attenuation coefficient exploration via Time-based Decay, Cuckoo Search Optimization, Decay Model Selection, and Content-based Similarity integration with TCC through item similarity scores.

The dissertation demonstrates substantial progress in temporal collaborative filtering through detailed theoretical analysis and practical validation using three real-world datasets: two from Amazon and one from MovieLens. The results show that the proposed approaches significantly improve the accuracy of the recommendation system compared to current models. By incorporating time-aware adjustments, the study highlights measurable enhancements in recommendation accuracy, underscoring the effectiveness of dynamic, time-based approaches in personalizing recommendation systems. These findings have important implications for future research and practical implementations in personalized recommendation services.

**Keywords:** Dynamic Attenuation Coefficient, Collaborative Filtering, Personalized Recommendations,

# PREFACE

This master's thesis marks a pivotal achievement in my academic journey, encapsulating the research and efforts undertaken throughout my studies. I would like to express my deepest gratitude to my advisor, Konstantinos Stefanidis, whose invaluable guidance and unwavering support have profoundly influenced both my research and writing. Their expertise and encouragement have been instrumental in helping me navigate the complexities of this work.

I would also like to extend my heartfelt appreciation to Zheying Zhang, whose assistance and insightful feedback were essential in refining my ideas and elevating the overall quality of this thesis. Furthermore, I wish to acknowledge the steadfast support of my family and friends, whose belief in my abilities has motivated me to overcome challenges and pursue excellence.

It is my hope that this work contributes meaningfully to the ongoing academic discourse and inspires further research and inquiry. Thank you for taking the time to engage with this thesis; I hope you find it both insightful and thought-provoking.

Tampere, 8 October 2024

Ainaz Ebrahimi

# CONTENTS

# LIST OF FIGURES AND TABLES

# LIST OF ABBREVIATION

| | |
|---|---|
| IoT | Internet of Things |
| TCCF | Time Correlation Coefficient Collaborative Filtering |
| TCC | Time Correlation Coefficient |
| CF | Collaborative Filtering |
| SVD | Singular Value Decomposition |
| RS | Recommendation System |
| MF | Matrix Factorization |
| KNN | K-Nearest Neighbour |
| T-UCF | Time-weighted User-based Collaborative Filtering |
| CE | Collaborative Evolution |
| CTR | Collaborative Topic Regression |
| CDL | Collaborative Deep Learning |
| BPTF | Bayesian Probabilistic Tensor Factorization |
| TSCIA | Time-Sensitive Collaborative Interest Aware |
| DTDM | Dynamic Time Drift Model |
| STG | Session-based Temporal Graph |
| TMF | Temporal Matrix Factorization |
| CSK-means | Cuckoo Search K-means |
| PCC | Pearson Correlation Coefficient |
| NLP | Natural Language Processing |
| MAE | Mean Absolute Error |

# 1. INTRODUCTION

## 1.1 Background

Recommendation technology is one prominent application of the many improvements that have been incorporated into IoT environments as a result of network technology's rapid advancement (Cui et al. 2020). In essence, this technology contributes significantly to Internet of Things services (Ranjan et al. 2018). By analyzing data about items, users, and their interactions, recommender systems can determine which products or services would be most appealing to a certain user. This software can be used by individuals or businesses (Bobadilla et al. 2013a). By selecting the most pertinent facts and services from a large amount of data, these systems aim to reduce information overload by providing individualized services. One of the primary features of recommendation systems is their ability to deduce a user's preferences and interests from their behavior, as well as that of other users, in order to offer tailored recommendations (Resnick and Varian 1997). Collaborative filtering (CF) (Schafer J. Benand Frankowski 2007), content-based (CB) (Pazzani Michael J. and Billsus 2007), and knowledge-based (KB) (Burke 2002) techniques are popular approaches for recommendation. Because they work so well, traditional methods like content-based filtering and collaborative filtering are still in high demand. Content-based filtering suggests products comparable to those the user has already expressed interest in; collaborative filtering is based on the tastes of similar users. Every strategy offers a unique set of benefits and drawbacks. For instance, content-based filtering may produce recommendations that are unduly specialized, on the other hand, issues with data sparsity, scalability, and the cold-start problem may arise with collaborative filtering (Adomavicius and Tuzhilin 2005; Schafer J. Benand Frankowski 2007). Nonetheless, there are ways to somewhat lessen these difficulties. When dealing with the difficulties linked to IoT, two main factors need to be taken into account. IoT service systems manage large amounts of intricate and constantly growing data (Cui et al. 2020). The data is usually characterized by high dimensionality, sparsity, rich content, and structural complexity, which renders ordinary data processing approaches inadequate. Moreover, the exponential growth in the assortment and volume of products compels buyers to dedicate additional time to finding certain items, frequently having to sift through irrelevant information, resulting in a time-consuming and tiring process. This arduous procedure has the potential to result in a substantial decline in clients (Cui et al.

2020). Another difficulty arises from the changing and developing nature of user interests. Conventional recommendation techniques frequently overlook the dynamic aspect of user preferences, which evolve over time due to changing interests, novel experiences, and seasonal patterns. Conventional systems assume that user interests remain constant, disregarding the dynamic relationship between user preferences and time. Users' interests are always changing; thus, a recommendation system that can accurately anticipate and address their immediate requirements is essential. In response to this issue, time-aware recommendation systems have been created, including temporal data to provide more accurate and pertinent recommendations.

An intriguing approach in this field is the collaborative filtering algorithm (CF) is proposed, which is based on the time correlation coefficient (TCC) and CSK-means (TCCF).(Cui et al. 2020) The TCCF algorithm groups people with similar preferences together in order to deliver swift and precise recommendations. Nevertheless, the existing implementations of TCC have constraints in accurately capturing the intricacies of temporal dynamics.

## 1.2 Problem Statement

While contemporary time-aware recommendation algorithms demonstrate a degree of efficiency, they are not without their shortcomings. Time-based Recommendation Systems, particularly those employing the Time Correlation Coefficient (TCC) approach, exhibit a commendable level of effectiveness. However, they frequently fall short in precisely capturing the nuanced and evolving nature of user preferences. The inability of these models to consistently track and adapt to the shifting patterns of user interests over various timeframes, or to adequately respond to distinct temporal trends, results in recommendations that lack accuracy. This deficiency can be attributed, in part, to the lack of robust and relevant parameters within the TCC framework that can effectively mirror the fluctuations in user preferences. Moreover, these models tend to neglect the aspect of item similarity, an element that is vital for the generation of precise rating predictions and warrants integration into the algorithmic structure. To surmount these challenges, it is imperative to refine and advance these models by embedding temporal dynamics and the fluidity of user preferences into the recommendation mechanism, guided by well-defined criteria. Such considerations prompt the following inquiries:

- How can the precision of time-aware recommendation systems, particularly those based on the Time Correlation Coefficient (TCC), be enhanced to more accurately reflect the dynamic evolution of user preferences?
- What modifications can be made to the existing TCC formula to better incorporate factors such as item similarity and temporal dynamics?
- Which innovative techniques can be developed to integrate temporal context more effectively, thereby increasing the accuracy and relevance of recommendations?

Therefore, the main goal of this dissertation is to enhance the accuracy of Time Correlation Coefficient (TCC) models through the introduction of refined algorithms and the deployment of four cutting-edge methodologies. These advancements are intended to more effectively assimilate temporal data, thus elevating the precision and applicability of the recommendations provided. The overarching aim is to augment user experience and interaction by delivering recommendations that are in sync with the dynamic nature of user preferences as they change over time.

## 1.3 Research Approach

The primary goals of this research study are as follows:

1. **Analyze Limitations**: Conduct a critical evaluation of the limitations inherent in existing Temporal Collaborative Filtering (TCC) models, focusing on their ability to accurately capture temporal user preferences.
2. **Propose a Modified TCC Formula**: Develop an enhanced TCC formula that more precisely reflects the influence of time on user preferences. This will involve introducing novel parameters and concepts to improve the model's temporal dynamics. Which includes different approaches:

    - Enhanced current TCC model through Content-based Similarity approach
    - Dynamic Attenuation Coefficient Exploration using Time-Based Decay
    - Dynamic Attenuation Coefficient Exploration using Cuckoo Search Optimization
    - Dynamic Attenuation Coefficient Exploration through Decay Model Selection

3. **Evaluate Effectiveness**: Validate the effectiveness of the proposed modifications and methods through rigorous theoretical analysis and comprehensive experimental evaluations using real-world datasets.

The research objectives focus on identifying and addressing limitations in existing TCC models, particularly their difficulty in accurately capturing the dynamic evolution of user preferences. To overcome this, the study proposes modifications like incorporating item similarity and dynamic attenuation coefficients through methods such as Time-Based Decay, and Cuckoo Search Optimization. These innovations specifically target the challenges of handling temporal shifts and adapting to evolving user behaviors. By validating these modifications through theoretical analysis and real-world testing, the study addresses the limitations and enhances the model's ability to provide more accurate and relevant recommendations over time.

## 1.4 Significance of the Study

Integrating temporal context into recommendation systems is essential for enhancing their accuracy and relevance. This study significantly advances the field through the following contributions:

- **Advancing Temporal Understanding:** Provides a comprehensive analysis of temporal dynamics in user preferences, offering insights into how these can be effectively modelled within recommendation systems.
- **Enhancing Recommendation Accuracy:** Introduces improved methodologies that yield more precise and timely recommendations, thereby increasing user satisfaction and engagement.
- **Broad Applicability:** Develops robust methodologies applicable across a variety of domains requiring personalized recommendations, including online retail, streaming services, and social media platforms.
- **Contributing to Research:** Offers novel models and algorithms that lay a strong foundation for future research in the area of time-aware recommendation systems, fostering further advancements in the field.

These contributions underscore the importance of temporal context in refining recommendation systems and highlight the potential for significant improvements in user experience across multiple application areas.

## 1.5 Structure of the Thesis

This thesis is organized as follows:

- **Chapter 2:** Literature Review
    - Provides an in-depth review of the current state of recommendation systems, focusing on time-aware approaches and the role of temporal context in user preferences.
    - Examines existing methods and algorithms, highlighting their strengths and limitations.
- **Chapter 3:** Theoretical Foundation
    - Explains the theoretical background of the TCC model, including its mathematical formulation and the rationale behind its development.
    - Discusses the identified limitations of existing TCC approaches and the need for modifications.
    - Introduces the proposed modifications to the TCC formula, detailing the new parameters and their roles in capturing temporal dynamics.
    - Provides an algorithmic explanation of offered methods.
- **Chapter 4:** Experimental Setup
    - Discusses the evaluation metrics used to assess the performance of the proposed methods.
    - Provides a description of datasets, and also Python libraries and specific packages are used in writing algorithms.
- **Chapter 5:** Experimental Results
    - Applies the proposed methods to real-world data to demonstrate their practical applicability and effectiveness.
    - Presents results, insights, and feedback to validate the findings.
- **Chapter 6:** Conclusion and Future Work
    - Highlights the study's contributions to recommendation systems and summarises its main findings.

    - Discusses the limitations of the study and suggests directions for future research.
- **References:**
    - Lists all the academic sources and references cited throughout the thesis.

# 2. LITERATURE REVIEW

## 2.1 Recommendation Systems

Recommendation systems are designed to provide suggestions to new users in any domain (Cai et al. 2024). The main goal of such systems is to provide pertinent suggestions to inexperienced users concerned with services or items that correspond to their interests. Recommender systems utilize various data mining approaches to identify commonalities among different data items (Chen and McLeod 2006). Instances of recommendation systems in practice include Amazon's algorithmic book recommendations and Netflix's personalized movie selections. The design of these systems is customized to suit the particular field and the unique characteristics of the data that is accessible. In order to provide suggestions, these systems analyze both product information and user data, generating a user-item similarity matrix that quantifies the similarity between users. Subsequently, this similarity data is utilized to produce recommendations for novel consumers. Typically, these systems utilize either collaborative filtering or content-based filtering strategies (Sayyed, Argiddi, and Apte 2013).

## 2.2 Content-Based Filtering

The content-based method is a specialized algorithm that analyzes the qualities of items in order to provide predictions. It is very efficient when suggesting materials such as web pages, publications, and news stories. Content-based filtering involves making suggestions by analyzing user profiles and extracting information from the content of things that users have already rated (Bobadilla et al. 2013a; Burke 2002). Subsequently, recommendations are made for items that bear resemblance to those that have received favorable ratings from the user.

Content-based filtering utilizes diverse algorithms to assess the similarity between papers, thereby producing pertinent recommendations. It can utilize the Vector Space Model, such as Term Frequency-Inverse Document Frequency (TF/IDF), or probabilistic models like the Naive Bayes Classifier (Friedman, Geiger, and Goldszmidt 1997), Decision Trees, or Neural Networks (Bartocci et al. 2018) to represent the connections be-

tween documents in a collection. These strategies produce recommendations by acquiring knowledge about underlying trends using statistical analysis or machine learning methods.

An inherent benefit of content-based filtering is its independence from other users' profiles, as their choices have no impact on the recommendations. In addition, the content-based filtering technique has the ability to promptly adjust its recommendations when there are changes in a user's profile. Nevertheless, a significant limitation is the requirement for a comprehensive comprehension and precise delineation of the characteristics of the elements inside the user profile (OpenAI 2024).

## 2.3 Collaborative Filtering

The most often used way to develop recommendation systems is collaborative filtering. Collaborative Filtering (CF) approaches are important in the recommendation process, and they are typically used in conjunction with other filtering strategies such as content-based and knowledge-based methods (Lops Pasqualeand de Gemmis 2011). Collaborative filtering methods rely on collecting and analyzing a substantial amount of data related to a user's behavior, activities, or preferences(Rostami et al. 2023). These methods then predict the preferences of the user by comparing their similarities with other users (Luo, Xia, and Zhu 2013; Schafer J. Benand Frankowski 2007). A significant advantage of the collaborative filtering approach is that it may provide precise suggestions for composite items without relying on machine-decomposable messages. Collaborative filtering recommendation systems offer items by analyzing the past ratings of a wide group of users. One of the pioneering systems that employed collaborative filtering, which utilizes user preferences to suggest items. There is a study that was the first to build an open architecture for filtering Usenet news, thus pioneering this approach. This technology established the foundation for contemporary collaborative filtering methodologies (Resnick et al. 1994).

Multi-domain collaborative recommender systems have been employed in several fields. GroupLens is a news-based system that uses collaborative ways to help users find items from a large news database (Adomavicius and Tuzhilin 2005b). Ringo is an internet-based system that employs collaborative filtering to construct user profiles by analyzing their ratings of music albums (Chen et al. 2008). Amazon employs subject diversity techniques to enhance its recommendation system (Ziegler et al. 2005). Collaborative filtering, a technique used in recommendation systems, can be classified into two main categories: memory-based and model-based (Bobadilla et al. 2013b; Breese, Heckerman, and Kadie 2013.).

## 2.4 Memory Based Techniques

The user's previously rated items play a vital role in finding comparable users, sometimes known as "neighbors," who have similar preferences (Zhao and Shang 2010; Zhu, Ye, and Gong 2009). After identifying these neighbors, several techniques can be employed to merge their tastes and produce suggestions. These approaches have shown significant efficacy, resulting in their widespread utilization in practical applications. Memory-based collaborative filtering can be implemented using either user-based or item-based methods.

- **User-Based Collaborative Filtering**

Through the assessment of users' evaluations of shared objects, user-based collaborative filtering calculates the degree of similarity between users. Then, using a weighted average of ratings from users who are comparable to the active user—the weights being the similarity scores between these users and the target item—it predicts the rating of an active user for an item (Isinkaye, Folajimi, and Ojokoh 2015). For this approach, a thorough mathematical framework has been created that addresses scalability and similarity calculation concerns (Herlocker et al. 2004).

- **Item-Based Collaborative Filtering**

Item-based filtering techniques focus on predicting user preferences by comparing item similarities rather than user similarities. The approach involves creating a model that captures the similarities between items. This process starts by extracting all items that an active user has rated from the user-item matrix. The next step is to evaluate how similar these items are to the target item. The algorithm then selects the top k items that are most similar to the target item and calculates their corresponding similarity scores. The final prediction is derived by computing a weighted average of the active user's ratings on these k similar items. By precomputing item similarities, this method significantly enhances both scalability and recommendation accuracy, which is particularly beneficial for systems with a large number of users (Sarwar et al. 2001). Recommendation system techniques is briefly shown in Figure 2-1.

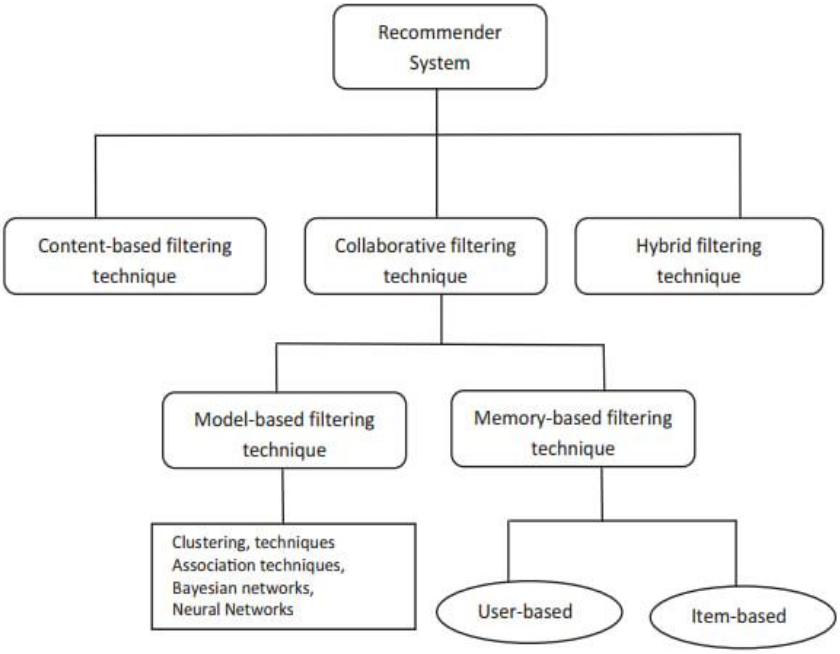


***Figure 2-1*** *recommendation system techniques (Isinkaye et al. 2015)*

**Similarity Measurment**

Cosine-based metrics and correlation-based metrics are two of the most often used similarity metrics. The definition of the Pearson correlation coefficient, which is used to determine how much two variables are linearly related, is as follows (Jannach 2010).

$$s(a,u) = \frac{\sum_{i=1}^{n}(r_{a,i}-\overline{r_a})(r_{u,i}-\overline{r_u})}{\sqrt{\sum_{i=1}^{n}(r_{a,i}-\overline{r_a})^2}\sqrt{\sum_{i=1}^{n}(r_{u,i}-\overline{r_u})^2}} \qquad (2.1)$$

Based on the given equation, n represents the total count of items in the user-item area, $\overline{r_a}$ denotes the average rating given by user a, and $s(a,u)$ signifies the similarity between two users, a and u. Furthermore, the weighted departure from the mean of the neighbors, which is determined by combining the ratings from the selected neighbors, is utilized to make predictions about an item. The fundamental equation for prediction is:

$$p(a,i) = \overline{r_a} + \frac{\sum_{i=1}^{n}(r_{u,i}-\overline{r_u})\times s(a,u)}{\sum_{i=1}^{n} s(a,u)} \qquad (2.2)$$

Instead of utilizing a statistical approach, cosine similarity functions are implemented inside a vector-space model that is grounded in linear algebra. This distinguishes it from the Pearson-based measure. It examines the arc formed by two n-dimensional vectors to quantify their similarity.

In information retrieval and text mining, this cosine-based metric is widely utilized for comparing text documents, which are represented as term vectors. According to (Adomavicius and Tuzhilin 2005b, Herlocker et al. 2004, andJannach 2010), the similarity between two elements, u and v, is defined as follows:

$$s(\vec{u},\vec{v}) = \frac{\vec{u}\cdot\vec{v}}{|\vec{u}| * |\vec{v}|} = \frac{\sum_i r_{u,i} r_{v,i}}{\sqrt{\sum_i r_{u,i}^2} \times \sqrt{\sum_i r_{v,i}^2}} \qquad (2.3)$$

## 2.5 Model-Based Techniques

This method builds a model to improve the efficacy of collaborative filtering techniques using ratings from the past. One option is to use data mining or sophisticated machine learning techniques to carry out the model creation process. Because these techniques rely on pre-computed models, they may recommend a list of items quickly, and their efficacy has been regularly shown to be on par with neighborhood-based recommender systems. These methods include matrix completion techniques, latent semantic analysis, regression and clustering methods, and dimensionality reduction approaches like Singular Value Decomposition (SVD) (Isinkaye et al. 2015). Model-based algorithms critically analyze the user-item matrix to identify latent correlations between items, enabling the construction and ranking of top-N recommendation lists. These methods significantly ameliorate the sparsity problems commonly found in recommendation systems. Furthermore, the application of learning algorithms has expanded the recommendation paradigm by providing guidance not only on the appropriate beverage choices for users but also on the optimal timing for consumption. To enhance the precision and effectiveness of model-based recommender systems, it is imperative to explore unconventional learning techniques.

## 2.6 Cluster-Based Collaborative Filtering Models

Traditional recommendation systems face significant hurdles due to the vast, complex, and ever-expanding nature of big data (Shi, Larson, and Hanjalic 2014). Novel cluster-based collaborative filtering approaches have been devised to tackle these challenges

by minimizing the quantity of data points while preserving precise predictions (Salah, Rogovschi, and Nadif 2016; Zhou et al. 2018). When it comes to collaborative filtering, clustering is an essential preprocessing step that combines information from users who are similar to each other and improves recommendation performance. There has been a great deal of research on cluster-based collaborative filtering, including a range of approaches such as co-clustering, item-based clustering, and user-based clustering. In order to classify movies based on user ratings, for instance, Zahra et al. (2015) used K-means method. This allowed people to be grouped based on the groups of objects they had rated. In a similar vein, users can be categorized according to common interests and actions. Furthermore, a Multiclass Co-Clustering (MCoC) model was put forth by Bu et al. (2016). It clusters comparable users by utilizing user-item subgroups. By examining the complex interactions that exist between users and things, this model creates subgroups that improve the precision and granularity of the clustering procedure. According to simulation data, this subgroup-based strategy can considerably outperform conventional techniques in terms of suggestion performance.

By capturing more complex links within the data, these clustering algorithms improve overall suggestion quality in addition to addressing the scalability and sparsity challenges inherent in large datasets. Therefore, in the Big Data era, the incorporation of such sophisticated clustering algorithms is essential to the ongoing development and efficacy of recommendation systems.

## 2.7 Hybrid Approaches

To enhance system optimization and address limitations and challenges of pure recommendation systems, hybrid filtering approaches integrate multiple recommendation algorithms (Adomavicius and Zhang 2012; Stern, Herbrich, and Graepel 2009). Hybrid approaches work on the premise that a mixture of algorithms, each with its own advantages over the other, will yield suggestions that are more accurate and successful than a single algorithm (Schafer J. Benand Frankowski 2007). The shortcomings of a single strategy can be mitigated in a combined model by utilizing numerous recommendation strategies. The approaches can be merged through one of the following techniques: building a single recommendation system that integrates the two types of approaches; performing the algorithms independently and merging the results; or including some content-based filtering into the collaborative approach. A survey of hybrid approaches is conducted, demonstrating that hybrid approaches can address problems like cold starts and improve the precision of suggestions (Burke 2002).

## 2.8 Time-Aware Recommendations

A notable constraint of conventional recommendation systems (RS) is their incapacity to consider the dynamic character of user preferences over an extended period, potentially resulting in decreased suggestion precision (Ahmed et al. 2017). For example, while a user is younger, they might like animation movies, but as they get older, they might prefer action or drama movies. Therefore, it is critical to comprehend these changing user preferences in order to make accurate predictions about future behavior—a challenge that classic RSs find difficult. With the realization that user preferences are not constant but rather vary over time, time-aware RSs have become the subject of much recent research. These studies can be divided into several categories based on how they provide recommendations and rate predictions. The time-weighted collaborative filtering technique is a well-known strategy among the different time-sensitive techniques.

## 2.9 Common methods in time-aware RS

The following lists some of the most often used methods for suggestions that are time-sensitive (Ahmed et al. 2017; Reham Alabduljabbar, Alshareef, and Alshareef 2023):

- **K-nearest-neighborhood (KNN)**
  The k-nearest neighbor (KNN) method is extensively applied to address data sparsity issues. However, it has a significant limitation: it offers poor coverage for users who share goods with the current user, resulting in fewer ratings from the active user's neighbors. When the rating matrix is sparse, the predictions generated by neighbors tend to be mediocre. Nonetheless, KNN can be improved by integrating it with other variables such as baseline, latent factors, and time to better predict sparse rating scores (R Alabduljabbar, and Alshareef 2023).

- **Matrix Factorization (MF)**
  Matrix Factorization (MF) has become a leading approach in collaborative filtering because of its effectiveness in tackling data sparsity and cold-start challenges. MF integrates techniques like Singular Value Decomposition (SVD), baseline, and latent factors. It is proficient at identifying latent semantic components and managing scaleability issues. The MF process involves stages that utilize SVD, baseline, and latent factors to extract latent factors from a rating matrix, thereby simplifying the rating scores. SVD, a straightforward MF method, is used to extract latent feedback

between users and products. Additionally, KNN can be combined with matrix factorization to enhance its performance (R Alabduljabbar et al. 2023).

- **Time-SVD++** Time-SVD++ is a well-known and comprehensive technique of time-sensitive recommendations. It models user preferences and item attributes by incorporating multiple time bins and accounting for unexpected drifts, as evidenced by the Netflix Prize competition. Time-SVD++ also considers additional implicit feedback and temporal effects (Koren and Bell 2015). Among these, the time-SVD++ model stands out as a modern and widely adopted method for handling temporal dynamics.

- **T-UCF (Time-weighted User-based Collaborative Filtering)** This method employs an exponential decay formula to represent the diminishing relevance of older data over time (Su et al. 2015).

- **Collaborative Evolution (CE)**This time-aware matrix factorization technique uses temporal regression to handle the dynamic evolution of user latent vectors over time (Wan, Chen, and Yan 2021).

- **Collaborative Topic Regression (CTR)** CTR uses Latent Dirichlet Allocation (LDA) to merge item content, assuming a linear relationship between the item latent vector and item topic distribution (Hamzehei et al. 2019).

- **Collaborative Deep Learning (CDL)** CDL is a hierarchical Bayesian model that combines deep representation learning for content with collaborative filtering for a rating (feedback) matrix (Wang, Wang, and Yeung 2015).

- **GRU-based RNN** Gated Recurrent Units (GRUs) are used to capture long-term data, enabling the interpretation of the global dynamic evolution of item ratings. Additionally, several innovative algorithms and models contribute to the field of time-sensitive recommender systems. These include Bayesian Probabilistic Tensor Factorization (BPTF), a novel temporal approach called Pegasus, a temporal recommender system known as TeRec, Neural Ordinary Differential Equation (NODE), and Time-Sensitive Collaborative Interest Aware (TSCIA) (Ahmed et al. 2017; Reham Alabduljabbar et al. 2023; Bao and Zhang 2021; Chen et al. 2013; Xiong et al. n.d.; Xue et al. 2019a).

## 2.10 Pioneering Research in Time-Aware Recommender Systems

The first researcher who inquired into models based on probabilistic MF for TimeSVD++ approaches was Koren (2009). He used a latent component, time change, implicit feedback, item bias, and user bias in the neighborhood model. Additionally, according to a different study, the dynamic time drift model (DTDM) (R Alabduljabbar et al. 2023)employs a time factor matrix along with clustering to estimate how far users' interests moved within a class. Sun and Dong (2017), Chua, Oentaryo, and Lim (2013)developed discrete temporal factorization models which reinforce multiple latent states at various time steps by utilizing dynamic matrix factorization (DMF) approaches. The evolution of user actions over time is tracked using non-negative matrix factorization. Additionally, Xiang et al. (2010), introduced session-based temporal graphs (STG), defining time bins through session-based temporal graphs across multiple time bins. Li et al. (2019) developed a different model that merged the RS approach with forgetting functions and similarity. Time weights in the latent transition model have not been employed in many studies to date. Furthermore, the proposed ratings using a three-dimensional tensor (user, item, and time) reflect tensor factorization, such as the Bayesian probabilistic tensor factorization (BPTF) model (R Alabduljabbar et al. 2023, and Xiong et al. 2010). The model can learn the global evolution of latent features and ratings through utilizing the inner product of the latent component vector; but its sensitivity is not adequate to capture changes in preferences at the local level (Wangwatcharakul and Wongthanavasu 2020). In order to forecast alterations in user inclinations across time, two models have been provided: the temporal matrix factorization (TMF) and the Bayesian TMF (BTMF). The main concept is to employ a latent transition matrix to depict evolution, which represents the time-invariant aspects of users' temporal dynamics (Zhang and Wang 2016).

In one study, a type of recommender system responsive to both time and location is introduced within a single model. The research proposes two approaches for implementing time-sensitive recommender systems: implicit feedback, which utilizes a user's past behaviors, such as purchases or webpage clicks, to predict future actions, and explicit ratings, where timestamps are linked to user ratings. Additionally, three techniques are discussed for leveraging temporal information to enhance prediction accuracy: recency-based models, periodic context-based models, and models that explicitly incorporate time as an independent variable (Aggarwal 2016).

A new method is proposed for calculating temporal weights of various items, effectively reducing the weight of older data. Consumer shopping habits tend to fluctuate, and the same individual may exhibit different reactions to different products. The suggested approach differentiates between multiple item categories by employing clustering techniques. Additionally, it tracks changes in each user's interest in purchasing specific item clusters, applying a customized decay factor based on the user's previous purchase history. According to Ding and Li (2005), the findings demonstrate that the suggested strategy greatly improves item-based collaborative filtering's precision without adding more computing complexity.

Yet another suggested strategy introduces, a new Recommender System that incorporates Temporal Reliability and Confidence measurements (RSTRC). The primary objective is to account for the changing pattern of user preferences over time when evaluating these indicators. This method differs from prior studies by incorporating the time dimension into both reliability and confidence measurements, recognizing the dynamic character of user activity. This technique reflects the initial endeavor to integrate confidence measures with temporal reliability in the recommendation process. In order to assess the efficiency of user rating profiles, a proficient probabilistic approach is utilized (Ahmadian et al, 2022).

A recent study introduced the Feature Stability and Time Sensitivity (FSTS) method, a novel search technique that incorporates both time-sensitive parameters and stability variables. This approach aims to enhance recommendation accuracy by accounting for changes in user preferences over time while maintaining the stability of certain features, providing a more balanced and adaptive recommendation process. Coverage, popularity, recall, and precision were the four evaluation criteria used to test the FSTS algorithm on the MovieLens dataset. To combat user interest drift, the prediction method retrieves item features using the feature vector and the time-sensitive factor. As a result, the FSTS algorithm enhances performance and prediction algorithms with little time-related complexity. Nevertheless, the time-sensitive factor's dynamic change was lacking (Pang et al. 2020). A novel recommendation algorithm that takes time weights into account for different products was introduced in a study. According to Xue et al. (2019b), consumers' buying habits change with time. Therefore, it's important to lower the weight of earlier data.

The article "Personalized Recommendation System based on Collaborative Filtering for IoT Scenarios" by Zhihua Cui et al. (2020) discusses how traditional recommendation

algorithms are limited in the Internet of Things (IoT) context. The authors introduce a new recommendation model called TCCF (Time Correlation Coefficient and CSK-means), which groups like users together to provide fast and accurate recommendations. The paper's main contributions are: Introducing a time correlation coefficient to consider The time-related aspect of user choices. A refined K-means method is enhanced by optimizing it using Cuckoo Search (CS) to improve the effectiveness of clustering. The conducted experiments provide substantial evidence that the suggested model enhances recommendation accuracy by 5.2% when compared to the MCoC model, as observed in the MovieLens and Douban datasets. The study emphasizes the need to taking into account the temporal dynamics in user preferences and proposes a reliable method for providing prompt and precise recommendations in Internet of Things (IoT) scenarios.

Building on the foundation laid by the previous study, this thesis distinguishes itself by leveraging temporal information to model user preference behavior more effectively and accurately. My innovative recommendation framework swiftly estimates item similarity based on text analysis, enabling the generation of more accurate recommendations. Unlike traditional methods, our model integrates dynamic algorithms to detect interest drifts, ensuring that recommendations remain relevant over time. My work addresses several key challenges in the field of recommendation systems:

- **Temporal Information Utilization:** Traditional recommendation systems often fail to account for the evolving nature of user preferences. Our approach integrates temporal data, allowing the system to model how user interests change over time. This leads to more precise and timely recommendations, which better reflect the current preferences of users.
- **Enhanced Item Similarity Estimation:** By employing advanced text analysis techniques, our framework can quickly and accurately estimate item similarity. This capability is crucial for generating high-quality recommendations, especially in environments with large and diverse datasets.
- **Dynamic Algorithms for Interest Drift Detection:** Users' interests are not static; they evolve due to various factors such as trends, personal growth, and external influences. Our model includes dynamic algorithms that monitor and adapt to these interest drifts. This feature ensures that the recommendations remain relevant and personalized, providing users with content that aligns with their current interests.

The proposed method aims to enhance recommendation accuracy by integrating temporal information and advanced text analysis to address key limitations such as evolving

user preferences. By dynamically adapting to changes in behavior and improving item similarity detection, it seeking for more personalized, relevant recommendations. This approach tries to address existing challenges, offering a more tailored and precise user experience.

# 3. THEORETICAL AND MATHEMATICAL FOUNDATION

## 3.1 An overview of the TCCF

Most conventional forms of collaborative filtering (CF) use users' ratings as a basis for product recommendations. Interest drift is the term used to describe how preferences could shift over time due to factors including trends, mood swings, and social influences. There are occasions when a user may be hyper-focused on a small number of points. As a result, traditional CF models that rely on comprehensive historical data may inadvertently suggest nonsense. If we want to measure how similar users are over time in collaborative filtering, we may use a time correlation coefficient (TCC)-based improved Pearson Correlation Coefficient (PCC). After establishing user groups through the application of k-means clustering and calculating TCC, we use a user-based CF algorithm to provide personalized product recommendations to each unique user.

**Algorithmic Steps:**

- Pick out starting positions for clustering by randomly choosing k data items from the dataset.
- Ensure the best performance of cluster centers: revise the original cluster nodes using the clustering fitness function and the Cuckoo Search algorithm.
- Use K-means clustering and finalize the clusters by applying the K-means algorithm with the optimized centers.
- Calculate TCC and multiply it by all ratings on the selected cluster.
- Recommendation Procedure: User Base: Locate nearby users in the clusters who have access to past data, and then tailor product recommendations to their tastes. In order to help new users or those without enough data, we recommend the most popular goods.

The precision of recommendations is improved as they become more pertinent and timely. Due to their consideration of the user's time-related choices. The methodology is highly appropriate for large datasets, as clustering reduces computational complexity.

Recommendations are generated by analyzing the interests of closely linked users using a specific collaborative filtering algorithm within clusters. This enhanced collaborative filtering (CF) model addresses the limitations of previous methods by enhancing the accuracy and efficiency of recommendations. It achieves this by using time-aware components and clustering techniques.

## 3.2 Cluster-Based CF Model via K-means improved by Cuckoo Search

In the era of big data, typical collaborative filtering algorithms face challenges in delivering prompt recommendations due to the rapid expansion of information. When dealing with extensive datasets, the process of identifying user similarities by analyzing all user information is inefficient and fails to consider the strong similarity in interests among users who evaluate the same item. The consequence of this is a reduction in the precision of recommendations.

To address these issues, this work used the K-means algorithm to divide the Big Data problem into smaller, more efficiently handled jobs. This involves grouping like users together to improve the speed and precision of recommendations. Moreover, a temporal component is included to monitor variations in the degree of similarity in interests across a specific timeframe. Optimizing the selection of beginning components in the typical K-means technique can significantly impact its effectiveness, posing a tough optimization problem. In order to enhance the quality of clustering outcomes, we employ the Cuckoo Search method, an intelligent optimization tool, to optimize the initialization variables.(Cui et al. 2020). By doing so, it seeks to improve collaborative filtering (CF) recommendation systems, ultimately leading to more accurate and effective recommendations.

- **K-means algorithm**

  One popular clustering algorithm is K-means, which efficiently groups data points with comparable characteristics. The K-means algorithm follows these basic steps:

  1. To begin, pick k objects at random from the dataset to serve as the centers of the first clusters.
  2. Allocate each item to the closest cluster based on its distance from all center points.
  3. Re-estimate the cluster centroids using the updated assignments.
  4. Until the centroids settle into a stable position or until the set number of times has been done is achieved, repeat steps (2) and (3).

How well the K-means algorithm works is heavily reliant on the initial center points that are chosen. When outliers are picked, clustering results can be unsatisfactory due to improper initialization.

- **Cuckoo Search (CS) Algorithm**

An inherent optimization method, the Cuckoo Search algorithm takes its cues from the cuckoo bird's parasitism of its own brood. Its powerful search capabilities enable it to effectively solve a wide range of optimization problems. Here are the basic rules that the CS algorithm follows:

1. The cuckoo only lays an egg in a nest that it chooses at random.
2. Generations after that tend to keep the nests that produced the best eggs.
3. The cuckoo will set up a new nest using Lévy flights if the host bird discovers its egg (with a probability p a ∈ [0,1] pa∈ [0,1]).

The widespread spot modify equation in CS is:

$$\underline{x_i}(t') = x_i(t) + \alpha \cdot 0.01 \cdot L \cdot \Big(x_i(t) - p_g(t)\Big) \cdot r \qquad (3.1)$$

where $x_i$ (t′) and $x_i$ (t) are the places of the $i_{th}$ cuckoo at times t′ and t. α>0 indicates the proportion of weight of the step size. The letter L is a Lévy distribution randomized value. There is a random number generator that uses a Gaussian distribution with the parameters N (0,1) to generate the value r.

To identify local search (when a cuckoo's nest is observed):

$$x_1(t+1) = x_i(t') + \text{rand} \cdot \Big(x_k(t') - x_j(t')\Big) \qquad (3.2)$$

where $x_k$ (t ′) and $x_k$ (t ′) and are randomly selected spots of cuckoos at time t ′.

In this algorithm, the function of assessment for clustering quality is used as the fitness function. For each value of j from 1 to k, set $\mu_j$ as the centroid of cluster Cj. The following formula indicates the fitness function for the Cuckoo Search (CS) algorithm:

$$f(x) = \sum_{j=1}^{k} \sum_{x_i \in C_j} \|x_i - \mu_j\| \qquad (3.3)$$

where, the user's rating of an item is expressed by the vector $x_i$. The expression $\sum \| x_i - \mu_j \|$ signifies the data density for all $x_i \in C_j$. Here is how the CSK-means algorithm works: For the cuckoos' starting positions, randomly choose k data elements from dataset D. Applying the aforementioned fitness function (Eq. 3.3), optimize the initial cluster centers using the cuckoo search algorithm. The best clusters will be produced by running the K-means clustering algorithm.

## 3.3 Time Correlation Coefficient (TCC)

An advanced recommendation approach called Time Correlation Coefficient Collaborative Filtering (TCCF) uses the temporal aspect of user interactions to improve the relevance and precision of recommendations. This strategy is important since conventional collaborative filtering techniques typically make the assumption that user preferences are static, which might result in recommendations that are out of date or irrelevant if they don't reflect the user's current interests.

In cases where user preferences in an item fluctuate over time, this phenomenon is called interest drift. We want to identify neighbors in CF whose interests align with the interests of current users. In most cases, the user's preferences tend to have somewhat normal distribution. Consequently, the TCC can be expressed in the manner shown below:

$$tcc_i = 1 - \frac{1}{\sqrt{2\pi}\sigma}\left(1 - \exp\left(-\frac{\Delta t^2}{2\sigma^2}\right)\right) \qquad (3.4)$$

Here, $\Delta t$ stands for the change in time, $t_1$ for the most recent ranked item, and $t_i$ for the timeline of the $i_{th}$ ranked item. σ is the attenuation coefficient, which measures the rate at which interest in the item decays. Different $\sigma$ values relate to different models.

The TCC formula may simulate various patterns of a user's evolving interests over time by manipulating the parameter σ, which is based on the timestamps of their item rankings.

## 3.4 CF model based on TCC and CSK-means (TCCF)

In this concept, users that closely match the target user are considered more influential references than others. We provide a refined Collaborative Filtering (CF) model, called TCCF, that combines Time Correlation Coefficient (TCC) with CSK-means to provide efficient and accurate recommendations. Algorithm 3-1 provides a detailed description of the TCCF model.

There are several things that need to be fed into the model: the number of clusters (k), the number of items to suggest (N), the user-item rating matrix, and additional Cuckoo Search (CS) factors. The CSK-means method is used to group users at first. When a specific user already has data, the model finds others with similar interests and suggests things that their neighbors like. New users are instructed to recommend popular items. The formula used to calculate item popularity is as follows:

$$ItemPop_i = \frac{|U_i|}{\sqrt{\sum_{i \in I} |U_i|^2}} \tag{3.5}$$

$U_i$ represents the users who have been reviewed for item *i,* while I represent the entire collection of things.

1. *Input: seting: parameters setting; data: user-item raking matrix; u: recommended user;*
2. *Output: results: top-N recommended items;*
3. *(k, N)←Initialize(setting) ;*
4. *Clusters ← clusterUsers (data, k) by CSK-means;*
5. *if u ∈ Users then*
6. *neighbors ← getNeighbors (u, clusters);*
7. *results ← recommendItems (neighbors, data, N) as Eq. (2.2);*
8. *End*
9. *Else*
10. *results ← recommendItems (data, N) as Eq. (3.5);*
11. *End*
12. *Return(results\text{results}results);*

***Algorithm 3-1*** *TCCF model (Cui et al. 2020)*

## 3.5 Proposed Creative Approaches

As mentioned earlier, the existing Time Correlation Coefficient (TCC) model, while precise, has limitations in addressing the evolving nature of user preferences. Its primary shortcomings are the inability to effectively track diminishing interest over time and a lack of comprehensive consideration for item similarities, which are crucial for accurate predictions. This study aims to overcome these limitations by proposing novel enhancements to TCC models that better integrate temporal context into the recommendation process. Ultimately, the goal is to develop advanced recommendation systems that can adapt to shifts in user preferences, delivering more precise, relevant, and personalized recommendations while improving user satisfaction and engagement with the platform.

**Novel Concepts in TCC**

- **Dynamic Modification of User Preferences**: The Modified TCC continuously refines its weight by considering not just recent user interactions but also similar interactions from users with comparable preferences. This distinguishes it from existing models that primarily focus on the latest data. By incorporating item similarity, the system can more accurately reflect the preferences of users with similar tastes, recognizing that these preferences can fluctuate over time due to new experiences, evolving interests, or even long-standing habits. This dynamic adjustment ensures that the model remains responsive to the broader patterns of user behavior, enhancing the relevance and effectiveness of its suggestions. As a result, users receive recommendations that are not only timely but also aligned with their unique and changing preferences, ultimately leading to a more personalized experience.
- **Creative Algorithms for Interest Drift**: The Modified TCC employs various approaches and algorithms to calculate the attenuation coefficient, a critical parameter that quantifies the rate at which user interests change over time. Unlike current models, which typically utilize a single, uniform attenuation coefficient for all users, the Modified TCC customizes this coefficient for each individual user. This tailored approach allows the model to more accurately reflect the unique dynamics of each user's preferences. By utilizing suggested techniques to determine the user-specific attenuation coefficient, the Modified TCC effectively tracks and measures shifts in user interests as they evolve in response to new experiences, external influences, or changes in behavior. This capability is essential for maintaining the accuracy and relevance of recommendations. Consequently, by integrating a robust mechanism for measuring interest drift that is customized for every user, the Modified TCC en-

hances its overall reliability as a recommendation system. Users benefit from suggestions that are not only timely but also more aligned with their current preferences, leading to a personalized experience that fosters greater user satisfaction and engagement.

The TCC formula would usually define how the weights for temporal proximity are determined, maybe by employing decay functions or others to lessen the weight of earlier interactions in comparison to more current ones or the same weight for old and new ratings. Having a heuristic algorithm that can calculate the true amount of interest drift (attenuation coefficient) can lead to a more reliable recommendation system. As I said earlier, I have made a number of changes to improve customer satisfaction by making recommendations more accurate. In the following the proposed approaches will be introduced.

## 3.6 Content-Based Similarity model

This approach enables the system to adopt a comprehensive perspective on the user's preferences, rather than exclusively emphasizing recent interactions. It effectively weighs the impact of both historical and recent data to offer more precise and comprehensive advice. Indeed, it guarantees that previous evaluations maintain their influence in molding the suggestion process, particularly when they accurately reflect the user's long-lasting preferences. This, in turn, enhances the system's personalization and effectiveness. Therefore, here are proposed enhancements to the formulas utilized for the Time Correlation coefficient employed in ratings. The algorithm was initially designed to calculate the similarity between things using Natural Language Processing (NLP) approaches. To be more specific, descriptive attributes of products that have been reviewed by a user are used to determine the similarity between the most recently rated product and other products that have been evaluated at different times by the same user. Subsequently, modifications to the TCC formula will be contemplated based on the similarity score. The pseudocode of the algorithm is shown in Pseudocode 3-1, which is followed by a detailed breakdown of its steps below.

1. **Descriptive Feature Analysis:** Use NLP techniques to analyze the descriptive features of products (such as brand, material, name of product, size, or other considerable things) that a user has rated. Through techniques such as tokenizing, removing unrelated words, and calculating frequency same as TF-IDF.

2. **Similarity Score Calculation:** Calculate the cosine similarity between the feature vector of the most recently rated product $frecent$ and each previously rated product $fj$. The cosine similarity S is given by:

$$s(frecent, fj) = \frac{(frecent, fj)}{\| frecent \| \| fj \|} \quad (3.6)$$

where $frecent$ and $fj$ denotes the dot product and $\| frecent \| \| fj \|$ denotes the Euclidean norm of vector $frecent$ and $fj$.

3. **Threshold and Rating Check:**
   - If the similarity score is above a predefined threshold and the rating is higher than 4, then treat the rating as new and set the TCC coefficient to 1.
   - Otherwise: Multiply the TCC coefficient by the ratings if the similarity score is lower or the rating is less than 4.

The proposed approach enhances precision in recommendations by improving relevance through the consideration of item similarity and recent ratings, while also showcasing adaptability by aligning with users' changing preferences over time. Additionally, it ensures efficient recommendations by integrating NLP methods for similarity calculations, which boosts accuracy in identifying relevant items, and employs dynamic TCC adjustment to reflect users' current interests accurately. Overall, this method aims to deliver more precise and relevant recommendations, enhancing user satisfaction and engagement.

```
Initialize sigma to 1.0
Get the vector for the most recent title and genre:
latest_title_vector = title vector of the most recently   rated item
for user
latest_genre_vector = genre vector of the most recently rated item for
user

Initialize lists for storing similarity scores and TCCs:
        similarity_scores = empty list
        cluster_tccs_1[user_id] = empty list

    For each record from index 1 to the last in cluster_records for the
    user:
             Calculate title cosine similarity:
             title_cs   =   cosine_similarity(latest_title_vector,   cur
             rent_title_vector)

             Calculate genre cosine similarity:
             genre_cs   =   cosine_similarity(latest_genre_vector,   cur-
             rent_genre_vector)

             Compute similarity score as the average of title_cs and
             genre_cs
             Append similarity score to similarity_scores list

            Determine TCC:
                If similarity_score > 0.3 then
                    tcc = 1.0
                Else
                   tcc = calculate_tcc(cluster_records, current_index,
                   sigma)
                EndIf

            Append tcc to cluster_tccs_1[user_id]

        Add a 'similarity_score' column to cluster_records[user_id]:
            Set the first value to 1 and the rest to similarity_scores

        Compute 'new_rating_1' as the product of 'tcc' and 'rating' for
     each record:
            new_rating_1 = rating * corresponding tcc

        Find minimum and maximum values of 'new_rating_1':
            min_val = minimum(new_rating_1)
            max_val = maximum(new_rating_1)

        Normalize 'new_rating_1' to a scale between 1 and 5:
            If max_val equals min_val then
                If max_val > 5 then
                    Set all values of 'new_rating_1' to 5
                ElseIf max_val < 1 then
                    Set all values of 'new_rating_1' to 1
                Else
                    Set all values of 'new_rating_1' to max_val
                EndIf
            Else
                Apply min-max normalization on 'new_rating_1':
                new_rating_1 = 1 + ((new_rating_1 - min_val) * (5 - 1)
     / (max_val - min_val))
     EndIf
```

***Pseudocode 3-1*** *Pseudocode of Content-Based Similarity*

## 3.7 Proposed methods to dynamically gain optimal sigma

The attenuation coefficient, which is explained about it earlier, also known as interest decay, is a vital quantity to consider when evaluating the impact of time decay on a recommendation system. Hence, it is crucial to ascertain this parameter in a manner that yields a precise and accurate value, rather than relying on unreliable trial-and-error methods. This is particularly challenging when dealing with real-world scenarios and extensive datasets. However, employing this approach can prove highly useful. Here, I propose and offer algorithms that are based on the properties and content of datasets, specifically ones linked to patterns of change. Below, these algorithms will be explained in depth.

### 3.7.1 Time-Based Decay method

This approach employs an exponential decay model to compute a sigma value, which serves as a measure of the pace at which user interest diminishes over time. The method calculates the average decayed values of time differences between contacts to provide a sigma value. This sigma value can then be utilized to personalize recommendations in a time-sensitive manner. The approach guarantees that the recommendation system maintains its adaptability to changes in user behavior, hence improving its accuracy and relevance. In the following the steps of this algorithm will be described and you can see the provided Pseudocode 3-2.

1. **Time Differences Calculation:** The method starts by computing the time difference between each rating and the next one using the diff() function on the normalized timestamps (timestamp_norm). This helps in understanding how the time intervals between user interactions vary.
2. **Decay Factor:** A constant decay factor (0.5 in this case (is based on knowledge domain and the data)) is defined. This factor determines the rate at which user interest decays over time.
3. ***Exponential Decay:*** The method applies an exponential decay function to the time differences. This models the decrease in user interest as a function of time, with the decay factor controlling the rate of decay. The formula used is:

$$\sigma i = e - decay\text{-}factor \times time\text{-}diff i \qquad (3.7)$$

4. **Average Sigma Value:** The method calculates the mean of the sigma values obtained from the exponential decay to produce a single sigma value representing the overall decay for the user.
5. **Calculate TCC:** Put all parameter in the TCC formula and calculate and multiply it in all the ratings.
6. **Generate Recommendations**

The benefits of this approach include enhanced temporal sensitivity, as it incorporates time differences and exponential decay to accurately capture how a user's interest diminishes over time, resulting in more personalized recommendations. Additionally, the use of an exponential decay function allows for smooth and continuous modeling of interest decay, which is often more realistic than a linear decay. Furthermore, the method demonstrates robustness by having a fallback to a default value, ensuring reliable performance even in edge cases where the calculated sigma might be zero.

```
FUNCTION calculate_sigma_2(user_data):
      # Step 1: Calculate the time differences between each rating and   the next one
      INITIALIZE time_diffs AS the difference between consecutive elements in user_data['timestamp_norm']
    REPLACE any null values in time_diffs WITH 0

    # Step 2: Define the decay factor
    SET decay_factor TO 0.5

      # Step 3: Calculate sigma as the exponential decay of the time differences
    INITIALIZE sigma_values AS an empty list
    FOR each time_diff IN time_diffs:
          CALCULATE sigma_value AS the exponential of (-decay_factor * time_diff)
        ADD sigma_value TO sigma_values

    # Step 4: Calculate the mean of sigma_values
    SET sigma TO the mean of sigma_values

    # Step 5: Return sigma if it is not zero, otherwise return 1.0
    IF sigma IS NOT EQUAL TO 0:
        RETURN sigma
    ELSE:
        RETURN 1.0
```

***Pseudocode 3-2*** *Pseudocode of Time-Based Decay*

### 3.7.2 Decay Model Selection method

This approach incorporates both exponential and Gaussian decay functions to conduct a comprehensive analysis of the temporal evolution of user ratings. By employing a dual-model technique, we are able to analyze various patterns of user behavior, enabling us to determine whether ratings exhibit a rapid decline (typical of exponential decay) or a more steady decline (indicative of Gaussian decay). By utilizing these models, we can acquire a comprehensive comprehension of user involvement, contentment, and potential patterns in ratings over a period of time. Algorithm steps are given below, and pseudocode 3-3 is provided accordingly.

1. **Check for Sufficient Data**: Ensure there are at least two data points for model fitting.
2. **Fit Exponential Decay Model**:
    - Use the formula: $y=a\cdot e-bx$ *(3.8)*
    - Fit to the data and extract the decay parameter b (denoted as b_exp).
3. **Fit Gaussian Decay Model**:
    - Use the formula: $y=a\cdot e-((x-b)/c)2$ *(3.9)*
    - Fit to the data and extract the decay parameter b (denoted as b_gauss).
4. **Return Decay Parameters**:
    - Return both decay parameters (b_exp and b_gauss) for further analysis.
5. **Calculate TCC:** Put all parameter in the TCC formula and calculate and multiply it in all the ratings.
6. **Generate Recommendations**

The benefits of using decay functions are significant for understanding user behavior. By analyzing the decay patterns of user ratings, businesses can gain valuable insights into how quickly user satisfaction diminishes over time, enabling targeted interventions to maintain or boost ratings. The parameters obtained from decay models provide a quantitative basis for informed decision-making, allowing businesses to identify whether ratings tend to decline sharply or gradually, which helps tailor their strategies accordingly. Furthermore, insights into the evolution of user ratings empower companies to design more effective engagement initiatives. For instance, if ratings show a rapid decline, timely follow-ups or targeted improvements can be implemented to enhance user satisfaction.

Additionally, the decay parameters assist in forecasting trends, enabling organizations to proactively address potential declines in ratings before they occur.

```
FUNCTION calculate_sigma_6(user_data):
    # Convert 'date_time' column to datetime format
    date_time = CONVERT user_data['date_time'] TO datetime

    # Calculate differences in days between consecutive dates
    x_data = DIFFERENCE in days BETWEEN consecutive elements in
    date_time
    REPLACE any null values in x_data WITH 0

    # Calculate differences in ratings
    y_data = DIFFERENCE BETWEEN consecutive elements in
    user_data['rating']
    REPLACE any null values in y_data WITH 0

    # Filter x_data to include only positive differences
    x_data = FILTER x_data TO INCLUDE only elements GREATER THAN 0
    y_data = SELECT elements in y_data CORRESPONDING TO indices of
    x_data

    # Reset indices of x_data and y_data
    RESET indices of x_data
    RESET indices of y_data

    # Check if there is enough data to fit
    IF LENGTH of x_data IS LESS THAN 2:
        RETURN None, None

    # Try to fit exponential decay model
    TRY:
    params_exp = FIT exponential_decay MODEL TO x_data AND y_data
    WITH initial parameters (1, 0.1)
        b_exp = SECOND ELEMENT of params_exp
    EXCEPT:
        b_exp = None

    # Try to fit Gaussian decay model
    TRY:
    params_gauss = FIT gaussian_decay MODEL TO x_data AND y_data
    WITH initial parameters (1, 1)
        b_gauss = SECOND ELEMENT of params_gauss
    EXCEPT:
        b_gauss = None

    # Return the parameters from both models
    RETURN b_exp, b_gauss
```

***Pseudocode 3-3*** *Pseudocode of Decay Model Selection*

### 3.7.3 Metaheuristic optimization technique, Cuckoo Search

The pseudocode provided in Pseudocode 3-4 utilizes past user interaction data through the Cuckoo Search optimization technique to determine the optimal sigma (σ) value for each user. The steps are as follows:

1. **Data Preprocessing:** Load and preprocess the dataset to extract important information like user ID, item ID, rating, and timestamp.
2. **Population Initialization:** Create a population of sigma values within a given range to generate a diverse set of starting solutions for the optimization process.
3. **Lévy Flight:** Use Lévy flight steps to update sigma values, which provide a mechanism for both big exploratory and small exploitative steps, hence improving the search for optimal solutions.
4. **Implement a Fitness Function:** Evaluate the performance of each sigma value by dividing the data into training and testing sets, using a recommendation system, and computing performance metrics such as precision and recall. The fitness function f(σ) is given by:

$$f(\sigma)=precision(\sigma)+recall(\sigma) \qquad (3.8)$$

5. **Cuckoo Search Optimization**:
   Iteratively update the sigma values using the Lévy Flight and fitness evaluation, maintaining the best solutions while discarding inferior ones, with the possibility of exploring new areas.

6. **Calculate TCC:** Put all parameter in the TCC formula and calculate and multiply it in all the ratings.
7. **Generate Recommendations**

The optimization method enhances accuracy by ensuring that the sigma value is fine-tuned to improve performance measures such as precision and recall, resulting in higher overall recommendation accuracy. Additionally, this approach considers the changing nature of user interests, making the recommendation system resilient to shifts in user behavior and preferences over time. By employing this strategy, personalized and optimal sigma values can be generated for each user, significantly increasing the effectiveness and accuracy of the recommendation system.

```
Define cuckoo_search_optimization(initial_sigma, sigma_range,
    max_iterations=2, pop_size=20, pa=0.25):
    Initialize population:
        population = Call initialize_population(pop_size,
sigma_range)
        best_sigma = population[0]
        Print '(Init) sigma:', best_sigma
        best_fitness = Call fitness_function(best_sigma)

    For iteration in range(max_iterations):
        For i in range(pop_size):
            Update position by Lévy flights:
                step_size = Call levy_flight(1.5)
                new_sigma = population[i] + step_size * (popula-
                tion[i] - best_sigma)
                new_sigma = Clip new_sigma to sigma_range
                Print '(i:', iteration, ', pop: ', i, ') sigma:',
                new_sigma
                new_fitness = Call fitness_function(new_sigma)

            If new_fitness > best_fitness:
                best_sigma = new_sigma
                best_fitness = new_fitness

            Abandon some solutions with probability pa:
                If random number < pa:
                    population[i] = random uniform value in
sigma_range

    Return best_sigma
```

***Pseudocode 3-4*** *Pseudocode of cuckoo search*

# 4. EXPERIMENTAL SETUP

## 4.1 Datasets

In this study, I will utilize three datasets to assess the proposed recommendation methods: two Amazon product review datasets focused on **phones and accessories** with 20.8M ratings, 1.3M items, and 11.6M users and **video games** (Hou et al. 2024), containing 4.6M ratings, 137.2K items, and 2.8M users along with the **MovieLens** (Harper 2015) dataset which includes 20000263 ratings across 27278 movies. This diverse selection allows for a comprehensive analysis of recommendation algorithms across different domains.

The selected datasets are particularly suitable for recommendation systems for several reasons:

**Real-World User Data**: Each dataset contains actual user interactions, including ratings and reviews. This authentic data source is critical for training recommendation algorithms, as it reflects genuine user preferences and behaviors.

**Diverse Product and Content Types**: The datasets encompass a broad range of products (phones, accessories, video games) and media (movies). This diversity enables the evaluation of recommendation methods across different contexts, testing their adaptability and effectiveness in various scenarios.

**Rich Feature Sets**: The datasets provide multiple features that are essential for understanding user preferences. For the Amazon datasets, features like **rating**, **timestamp**, **user ID**, **ASIN**, and **parent ASIN** allow for tracking user behavior and product relationships. Additionally, metadata features such as **title**, **brand**, **material**, and **size** enhance the context for product recommendations. In the MovieLens dataset, features such as **rating**, **timestamp**, **user ID**, **movie ID**, **genre**, and **title** contribute to a well-rounded understanding of user preferences in film.

**Potential for Collaborative Filtering**: Both the Amazon and MovieLens datasets facilitate collaborative filtering techniques, which rely on user interactions to identify patterns

and make recommendations. The rich user-item interaction data allows the algorithms to find similarities between users and items effectively.

**Dynamic User Preferences**: The inclusion of timestamps in the datasets helps capture the temporal dynamics of user preferences. Understanding how tastes evolve over time is crucial for making timely and relevant recommendations.

**Selected Features for Evaluation:**

For the evaluation, I will focus on the following key features from the datasets:

- From the **Amazon Product Review Datasets**:

    1. **Rating (float)**: Reflects user satisfaction and serves as a primary indicator for recommendations.
    2. **Timestamp (int)**: Indicates when the review was posted, helping to analyze trends over time.
    3. **User ID (str)**: Allows tracking of individual reviewer contributions for collaborative filtering.
    4. **Parent ASIN (str)**: Groups products with variations, linking reviews to the correct product families.
    5. **ASIN (str)**: The unique identifier for the specific product.
    6. **Title (str)**: Provides context for the review.
    7. **Material, Brand, Size (from Details)**: Attributes that influence user choices and preferences.

- From the **MovieLens Dataset**:

    1. **Rating (float)**: User ratings for movies are crucial for generating personalized recommendations.
    2. **Timestamp (int)**: Time when the rating was submitted, aiding in trend analysis.
    3. **User ID (str)**: Facilitates collaborative filtering by tracking user interactions.
    4. **Movie ID (str)**: The unique identifier for each movie.
    5. **Genre (list)**: Helps categorize films and match them with user interests.
    6. **Title (str)**: Provides context for the ratings.

By utilizing these specific datasets, the evaluation aims to leverage their strengths to understand and predict user preferences effectively. The combination of real-world user data, diverse product types, and rich feature sets provides an ideal foundation for assessing the proposed recommendation methods. This structured approach will ensure that the evaluation is comprehensive, relevant, and insightful, ultimately contributing to the advancement of recommendation systems in various domains.

## 4.2 Libraries and Packages

### 4.2.1 Packages

In this study, a Python package is utilized to develop the recommendation system. The following sections will introduce this package, explain how it is used, and highlight its features. **Surprise** (Nicolas Hug 2017), is a Python library specifically designed for building and analyzing recommendation systems. Its full name, **Simple Python Recommendation System Engine**, reflects its purpose: to simplify the process of creating recommendation engines that can handle explicit rating data. The library is particularly useful for developers and researchers who want to implement collaborative filtering techniques, which are commonly used in recommendation systems.

**Key Features of Surprise**

1. **Ease of Use**: Surprise is designed to be user-friendly, making it accessible even for those who may not have extensive experience in machine learning or recommendation systems. It provides a straightforward API that allows users to quickly set up and experiment with different recommendation algorithms.
2. **Built-in Algorithms**: The library includes a variety of algorithms for collaborative filtering, such as user-based and item-based approaches, as well as matrix factorization techniques like Singular Value Decomposition (SVD). This variety enables users to choose the most suitable algorithm for their specific dataset and recommendation goals. However, in this study, the NMF algorithm is used because of the non-negative value.
3. **Data Handling**: Surprise can easily handle traditional datasets, such as the MovieLens datasets, which are commonly used for benchmarking recommendation systems. Additionally, it allows users to load their own datasets in various formats, including CSV files and Pandas DataFrames, making it versatile for different applications.

4. **Cross-Validation and Evaluation**: The library provides tools for cross-validation and performance evaluation, allowing users to assess the accuracy of their recommendation models. This is crucial for understanding how well a model will perform in real-world scenarios.
5. **Flexibility**: Users can customize their experiments by adjusting parameters and selecting different algorithms, giving them control over the recommendation process. This flexibility is essential for fine-tuning models to achieve optimal performance.

**How Surprise is used for Recommendation Systems**

Surprise can be utilized in several steps to build an effective recommendation system:

1. **Data Preparation**: Users can load their datasets into Surprise, which can include user-item interactions in the form of ratings. The library supports various data formats, making it easy to integrate with existing data pipelines.
2. **Choosing an Algorithm**: Based on the nature of the data and the specific requirements of the recommendation task, users can select from a range of built-in algorithms. For example, if the dataset is large and sparse, matrix factorization techniques like SVD may be appropriate.
3. **Training the Model**: Once an algorithm is selected, users can train the model using their dataset. Surprise handles the underlying computations, allowing users to focus on the higher-level aspects of model development.
4. **Making Predictions**: After training, the model can be used to make predictions about user preferences for items they have not yet rated. This is the core functionality of a recommendation system, as it provides personalized suggestions to users.
5. **Evaluating Performance**: Surprise includes tools for evaluating the model's performance using metrics such as Mean Absolute Error (MAE) and F-measure. This evaluation helps users understand how well their recommendation system is performing and where improvements may be needed.
6. **Iterating and Improving**: Based on the evaluation results, users can iterate on their models by trying different algorithms, adjusting parameters, or refining their data. This iterative process is essential for developing a robust recommendation system.

### 4.2.2 Libraries

Here is a brief overview of the libraries utilized in the algorithms and proposed methods:

1. **NumPy (import numpy as np)**:

   NumPy is a fundamental package for numerical computing in Python. It provides support for arrays, matrices, and a wide range of mathematical functions. In recommendation systems, NumPy is often used for efficient data manipulation and mathematical operations, such as calculating distances or performing matrix operations.

2. **SciPy (from scipy.special import gamma)**:

   SciPy builds on NumPy and provides additional functionality for scientific and technical computing. The gamma function is part of the special functions module and can be used in statistical calculations. In recommendation systems, SciPy is useful for optimization and advanced mathematical computations.

3. **Scikit-learn (from sklearn.metrics.pairwise import euclidean_distances)**:

   Scikit-learn is a powerful machine learning library that provides tools for data mining and data analysis. The euclidean_distances function calculates the Euclidean distance between pairs of points, which is essential for many recommendation algorithms that rely on distance metrics to find similar items or users.

4. **TQDM (from tqdm import tqdm)**:

   TQDM is a library for creating progress bars in Python. It is particularly useful for tracking the progress of loops, especially when processing large datasets in recommendation systems.

5. **NLTK (import nltk)**:

   The Natural Language Toolkit (NLTK) is a library for working with human language data (text). It provides tools for text processing, including tokenization,

stemming, and tagging. In recommendation systems, NLTK can be used for natural language processing tasks, such as analyzing user reviews or comments.

6. **CountVectorizer (from sklearn.feature_extraction.text import CountVectorizer)**:
   Token counts are transformed into a matrix using CountVectorizer, a feature extraction approach, from a set of text documents. For machine learning models, such as recommendation systems, this is helpful for converting textual data into a numerical format.

7. **Curve Fit (from scipy.optimize import curve_fit)**:

   This function is used for fitting a curve to data points. In the context of recommendation systems, it can be applied to model user preferences or item characteristics based on historical data.

8. **Surprise (from surprise import Dataset, Reader, NMF, accuracy)**:

   Surprise is a specialized library for building recommendation systems. It provides tools for loading datasets, defining models (like non-negative Matrix Factorization, NMF), and evaluating their performance. This library is particularly useful for implementing collaborative filtering techniques.

9. **Train-Test Split (from surprise.model_selection import train_test_split)**:

   This function is used to split the dataset into training and testing sets, which is crucial for evaluating the performance of recommendation algorithms. It helps ensure that the model is tested on unseen data.

10. **Cosine Similarity (from sklearn.metrics.pairwise import cosine_similarity)**:

    Cosine similarity measures the cosine of the angle between two non-zero vectors. It is commonly used in recommendation systems to determine the similarity between items or users based on their feature vectors.

11. **Collections (from collections import defaultdict)**:

    The defaultdict class from the collections module provides a dictionary-like object that provides default values for nonexistent keys. This can be useful for organizing data in recommendation systems, such as aggregating user preferences.

12. **Copy (import copy)**:

    The copy module allows for creating shallow or deep copies of objects. This can be useful in recommendation systems when manipulating data structures without altering the original data.

13. **Math (import math)**:

    The math module provides access to mathematical functions. It can be used for various calculations in recommendation systems, such as computing distances or performing statistical analyses.

14. **Optuna (import optuna)**:

    Optuna is an optimization framework for automating hyperparameter tuning. In recommendation systems, it can be used to optimize model parameters to improve performance.

15. **Matplotlib (import matplotlib.pyplot as plt)**:

    Matplotlib is a plotting library for creating static, animated, and interactive visualizations in Python. It can be used to visualize the results of recommendation algorithms, such as showing the distribution of user ratings or the performance of different models.

16. **Seaborn (import seaborn as sns)**:

    Seaborn is a statistical data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive statistical graphics, which can be useful for visualizing data distributions and relationships in recommendation systems.

## 4.3 Evaluation Metrics

To evaluate the performance of the recommendation methods, I will employ several metrics that provide insights into the accuracy and effectiveness of the predictions:

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:

$$R = \frac{|N(i) \cap M(i)|}{|M(i)|} \quad (4.1)$$

Here, N(i) is the number of recommended items to the target user u, and M(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:
   $$P = \frac{|N(i) \cap M(i)|}{|N(i)|} \quad (4.2)$$

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:

$$F - \text{ measure } = \frac{2 * P * R}{P + R} \quad (4.3)$$

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:

$$MAE = \frac{\sum_{i=1}^{N} |p_i - q_i|}{N} \quad (4.4)$$

Here, $p_i$ represents the predicted user scores, and $q_i$ represents the corresponding actual scores.

By applying these metrics—recall, precision, F-measure, MAP, and MAE—we can comprehensively assess the efficacy of our recommendation techniques. Each indicator offers unique perspectives on the system's effectiveness, ranging from the precision in finding pertinent objects to the overall excellence of recommendations. Comprehending this data is crucial for enhancing algorithms and guaranteeing that the recommendation system efficiently fulfills user requirements.

## 4.4 Time Complexity

- **Time Complexity**
  computing complexity is a mathematical notion in computer science that quantifies the computing time required for an algorithm to finish, based on the length of the input. This approach offers a means to assess the effectiveness of an algorithm by examining the correlation between the running time and the amount of the input.
- **Importance of Time Complexity**
  Understanding time complexity is crucial because it helps programmers and computer scientists predict how an algorithm will perform, especially with large inputs. This knowledge allows for better resource planning and ensures that the algorithm can handle the expected workload efficiently.

- **Time Complexity Measuring**
  Time complexity is commonly expressed using **Big O notation**, which provides an upper bound on the time an algorithm will take in the worst-case scenario. This notation helps in comparing the efficiency of different algorithms by focusing on their growth rates rather than exact execution times.

- **Examples of Time Complexity**

  **O(1)**: Constant time complexity, where the execution time does not change with the size of the input.

  **O(n)**: Linear time complexity, where the execution time grows linearly with the input size.

  **O($n^2$)**: Quadratic time complexity, where the execution time is proportional to the square of the input size. An example is a simple nested loop iterating over the input

**O(log n)**: Logarithmic time complexity, which is more efficient than linear time for large inputs. This is often seen in algorithms like binary search.

By understanding these concepts, developers can choose the most appropriate algorithms for their needs, ensuring optimal performance and resource usage.

# 5. EXPERIMENTAL RESULT

In this section, we analyze the results of our experiments, which were specifically crafted to assess the efficacy of the suggested methodologies when applied to diverse datasets and evaluated using various metrics.

## 5.1 Scatter Plots comparison

A scatter plot is a powerful data visualization tool that uses Cartesian coordinates to display values for two continuous variables, revealing relationships and correlations between them. In this context, scatter plots are used to compare actual user ratings with predicted ratings from various recommendation system methods. These graphs are crucial for evaluating the accuracy of a recommendation system. The closer the data points are to the diagonal line (running from the bottom left to the top right), the more accurate the predictions are. Below, we present scatter plots for different proposed methods along with their analysis, providing insights into their performance and accuracy.

### 5.1.1 MovieLens dataset

**Method 0 (TCCF):** Figure 5-1 illustrates a variety of predicted scores, with numerous points deviating significantly from the diagonal line. However, this method clusters more points along the line, indicating that the real scores and predictions are more closely aligned, suggesting increased accuracy. Still, additional metrics are necessary to conclusively determine the effectiveness of this method.

**Method 1 (Content-Based Similarity):** There is a noticeable dispersion, but the majority of points cluster along the diagonal, similar to Method 0. Figure 5-1 indicates some correlation between the predicted and actual ratings, yet significant errors remain. Although this method slightly differs from Method 0, it doesn't conclusively demonstrate consistency or accuracy.

**Method 2 (Time-Based Decay):** This method clusters more points along the diagonal, indicating that predictions align more closely with actual ratings, suggesting increased accuracy. Based on these scatter plots, this method performs similarly to Method 0, albeit slightly better.

**Method 3 (Optimized Sigma-Cuckoo Search):** This plot shows less effective clustering around the diagonal compared to Methods 0 and 1, and it includes some outliers. Although this approach appears to perform well, it doesn't achieve the same level of accuracy as Methods 0 and 2. The presence of outliers suggests variability in prediction accuracy.

**Method 4 (Decay Model Selection):** The points are closely packed together along the diagonal, which means the estimate is very accurate. There aren't many outliers, which suggests that this method does a good job of tracking user tastes over time. According to the results, this method works as well as the time-based and TCCF method and makes accurate guesses most of the time.

**Better method selection:**

The scatter plots of actual versus predicted ratings for the three methods—**time-based, decay model selection, and the TCCF method**—show similar patterns, with points generally aligning along the diagonal. This suggests that all three methods have comparable predictive performance when evaluated visually through these plots. However, to make a more informed decision about which method is superior, we will need to assess additional metrics beyond just the scatter plots. This further evaluation will help determine the most effective method in terms of accuracy and consistency. The scatter plots of all methods in MovieLens datasets are provided in Figure 5-1.

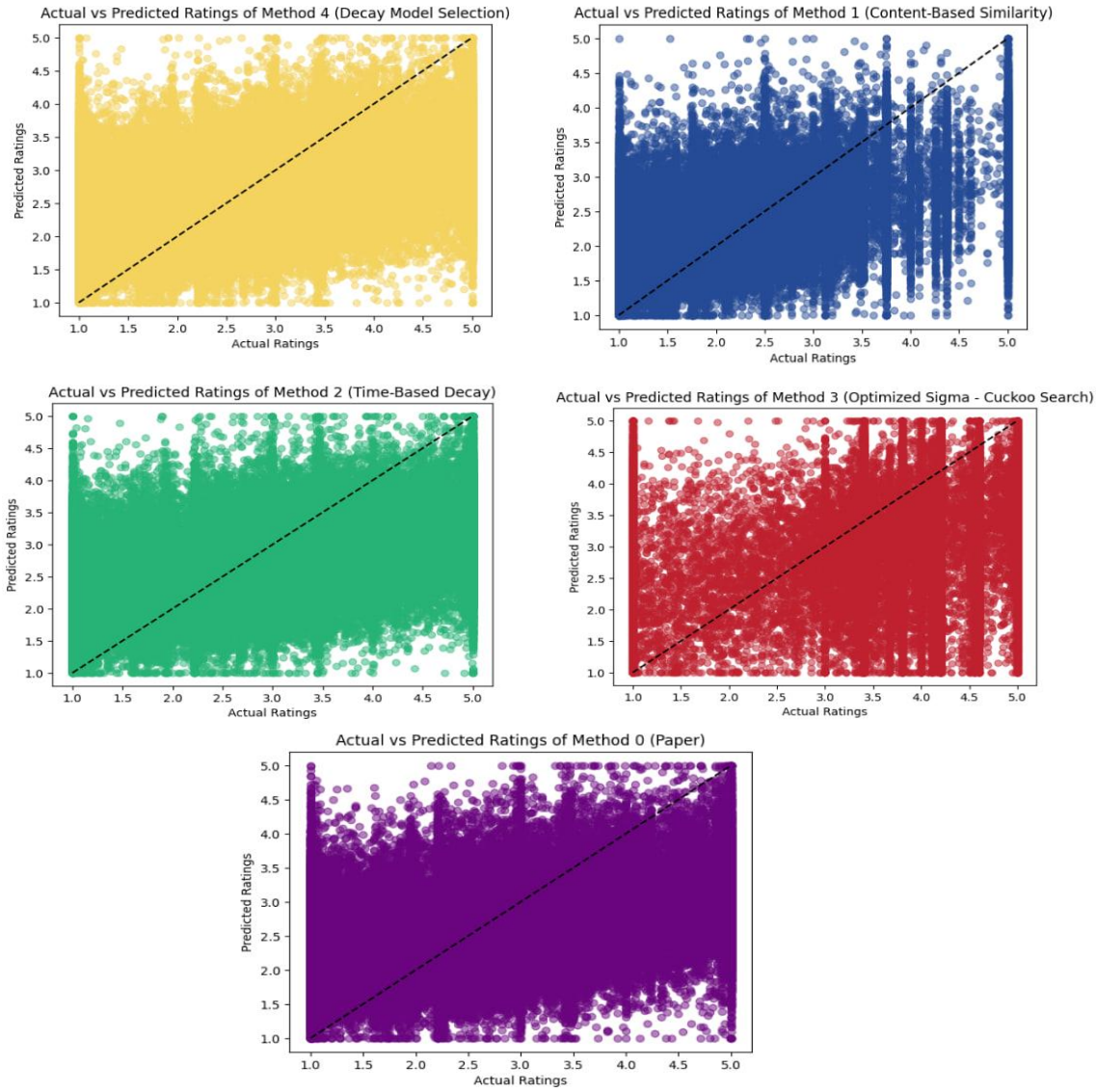


***Figure 5-1*** *Actual rating VS Predicted rating in MovieLens dataset*

## 5.1.2 CellPhone and Accessories dataset:

**Method 0 (TCCF):** Based on the given plot in Figure 5-2, There is a lot of variation around the 4-star rating, which means that many ratings are predicted right, but there are big differences, especially at lower ratings.

**Method 1 (Content-Based Similarity):** similar to method 0, This method seems to be in a different position in line with real scores, especially when ratings are at around 2 and 2.5. There are still groups at some points, though, which shows some accurate predictions.

**Method 2 (Time-Based Decay):** This method is like Method 1, but there is a clear spread at lower scores. At higher scores, it shows that it can make some predictions.

**Method 3 (Optimized Sigma - Cuckoo Search):** The plot demonstrates that this method outperforms Methods 0 and 1 due to its reduced dispersion. Compared to the previous methods, it can be considered more accurate.

**Method 4 (Decay Model Selection):** The scatter, indicates that this method has trouble making predictions across the board, especially at lower scores, which means it is less accurate. Like methods 0 and 1.

**Better method selection:**

While it's challenging to determine the most accurate method solely from these plots, at first glance, the time-based method, **cuckoo search**, and **content-based similarity** appear superior. Additional validation metrics will be used for further evaluation.

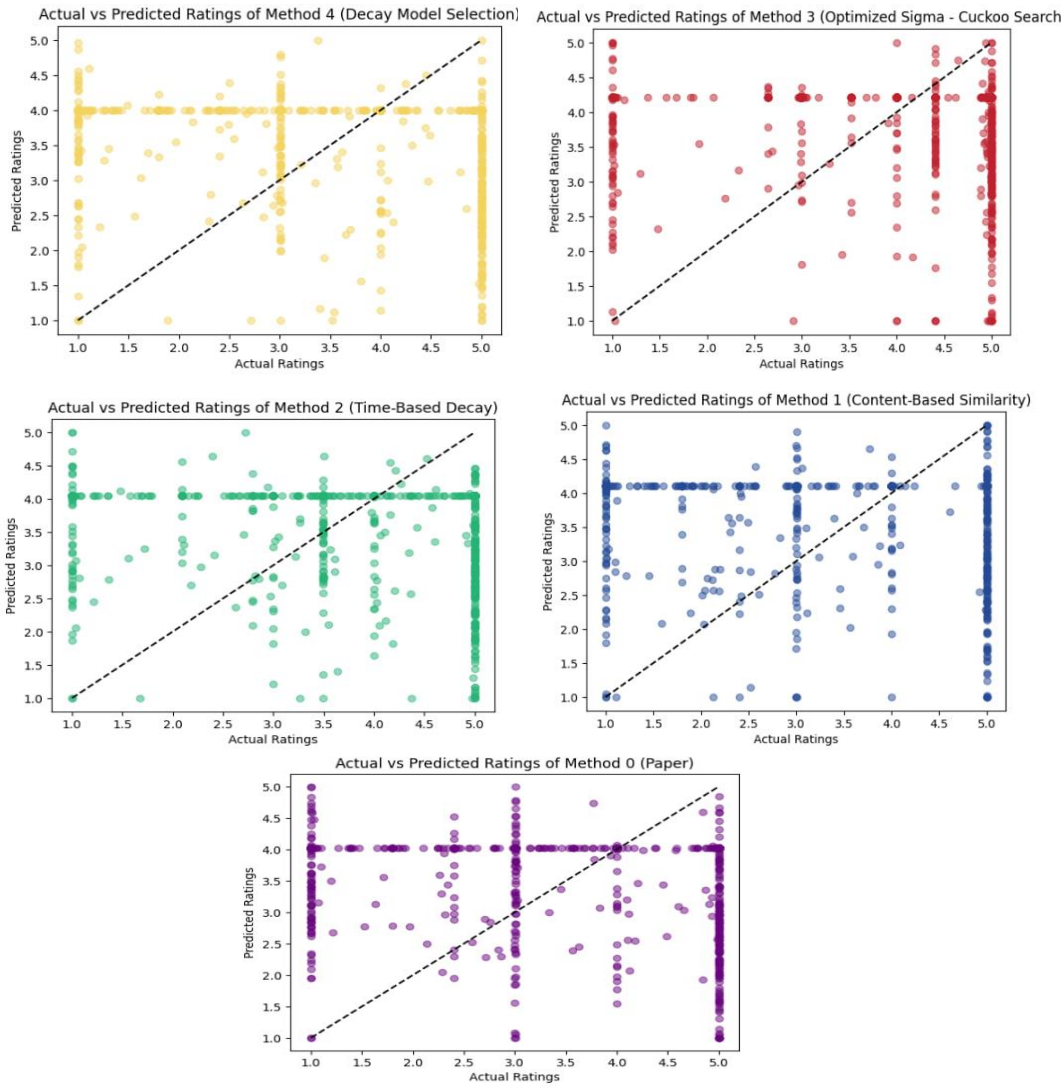


***Figure 5-2*** *Actual rating VS Predicted rating in CellPhone dataset*

### 5.1.3 Video Games dataset:

**Method 0 (TCCF):** The plot in Figure 5-3 shows a lot of different predicted scores, with many points very far from the diagonal line. There are numerous predictions based on all the real ratings, indicating they are not very accurate.

**Method 1 (Content-Based Similarity):** There is a clear spread, though some points are slightly grouped along the diagonal. This plot shows some connection between predicted and real ratings, but there are still significant errors. While slightly better than Method 0, it still doesn't provide consistent or accurate predictions.

**Method 2 (Time-Based Decay):** Here, more points are grouped together along the diagonal, suggesting predictions are more in line with real ratings, indicating higher accuracy. By considering timing, this method captures changes in user preferences better, leading to more accurate predictions.

**Method 3 (Optimized Sigma - Cuckoo Search):** This plot has better grouping around the diagonal than Methods 0 and 1, but still has some outliers. It performs better than the first two methods but isn't as accurate as Method 2.

**Method 4 (Decay Model Selection):** The scatter plot reveals that this method struggles to make accurate predictions overall. There is a lot of dispersion of points, which is far from the diagonal line.

**Better method selection:**

Overall, if we want to choose a better method based on the scatter plots from all three datasets, Method 1: **the Content-Based model**, Method 2: **the Time-Based model**, and Method 3: **the Cuckoo Search model** consistently exhibit the closest alignment with the diagonal, suggesting high accuracy and few outliers. To make a more informed decision, further evaluation using additional validation metrics will be conducted.

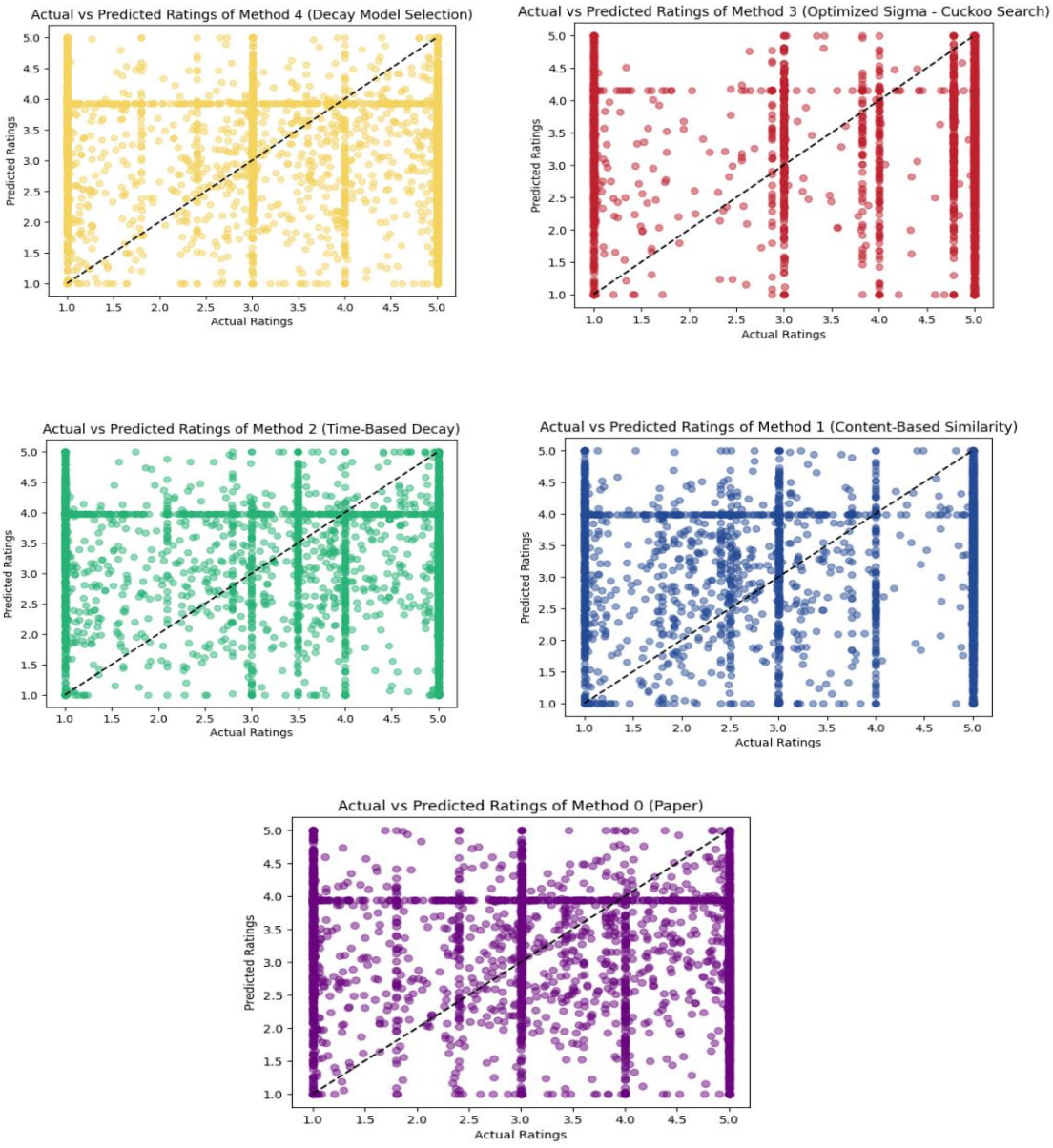


***Figure 5-3*** *Actual rating VS Predicted rating in VideoGames dataset*

## 5.2 Comparison methods: Metrics and Time Complexity

The initial step will involve introducing the time complexity of each method, as illustrated in Table 5-1, accompanied by relevant explanations. Subsequently, the results of various metrics—namely recall, F-measure, precision, and MAE—obtained from running three different datasets will be presented, and a comprehensive analysis of these results will follow.

***Table 5-1*** *Time complexity of different proposed methods*

| Method | Complexity | Where |
|---|---|---|
| Method 0 (TCCF) | O(n * m * j) | n: number of users, m: number of records for a user, j: trial step for reaching to optimized attenuation coefficient |
| Method 1 (Content-Based Similarity) | O(n * m * d) | n: number of users, m: number of records for a user, d: dimension of the vectors |
| Method 2 (Time-Based Decay) | O(n * m) | n: number of users, m: number of records for a user |
| Method 3 (Optimized Sigma - Cuckoo Search) | O(i * p * n * m) | n: number of users, m: number of records for a user, i: number of iterations, p: population size in the Cuckoo Search optimization |
| Method 4 (Decay Model Selection) | O(f * n * m) | n: number of users, m: number of records for a user, f: complexity of fitting the decay models (generally small but variable) |

The above table outlines five methods with their respective time complexities using Big O notation. Each method is analyzed based on variables like the number of users (n), records per user (m), and vector dimensions (d). The descriptions include specific parameters for each method, such as iterations in Cuckoo Search (i) and complexity of decay models (f). This breakdown helps compare the computational efficiency of different approaches.

### 5.2.1 Analysis of Methods in MovieLens dataset

The implementation of various proposed methods on the Movielens dataset yields significant results, as detailed in Table 5-2, which presents the values for recall, precision, F-measure, and MAE. Additionally, these results are depicted in a histogram to facilitate better understanding and easier comparison between the different approaches in Figure 5-4.

***Table 5-2*** *Metrics Value in MovieLens dataset*

| Metric | TCCF | Content-Based Similarity | Time-Based Decay | Cuckoo Search | Decay Model Selection |
|---|---|---|---|---|---|
| **MAE** | 0.825807 | 0.686407 | 0.823619 | 0.896408 | 0.820437 |
| **Precision** | 0.760404 | 0.872948 | 0.762204 | 0.777843 | 0.761306 |
| **Recall** | 0.311950 | 0.332900 | 0.327388 | 0.496334 | 0.324281 |
| **F-measure** | 0.286534 | 0.312209 | 0.300798 | 0.470321 | 0.298416 |

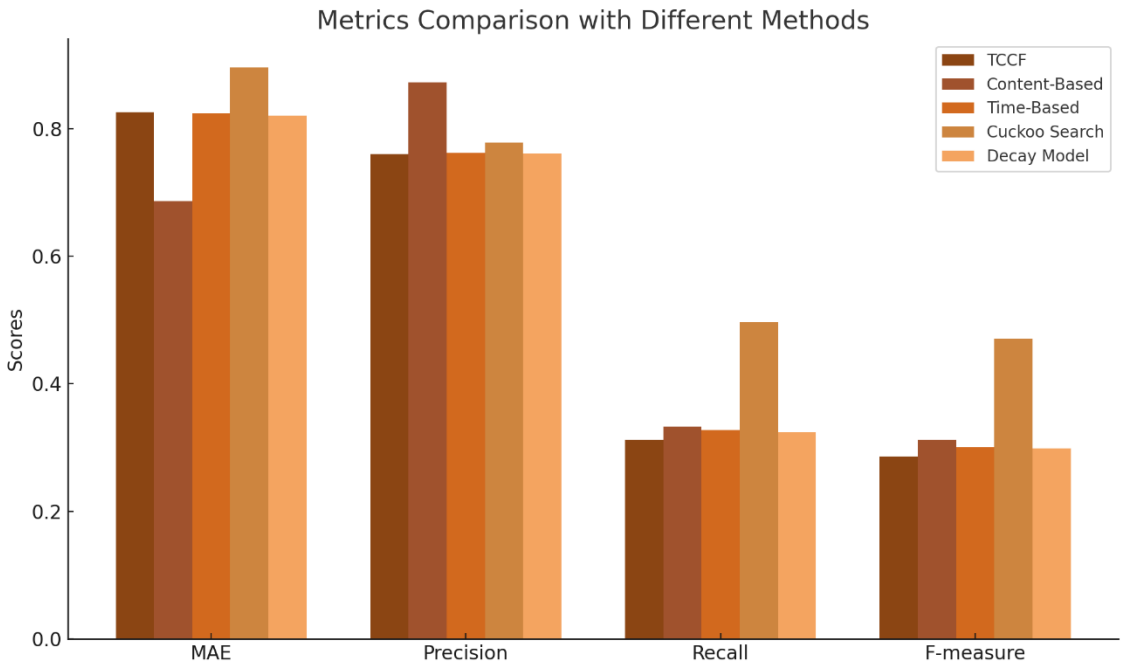


***Figure 5-4*** *Metric Comparison in MovieLens dataset*

**Content-Based Similarity:** This method excels in both MAE and Precision, with the lowest MAE (0.686407) and highest Precision (0.872948), indicating accurate and relevant recommendations. Its time complexity, $O(n \times m \times d)$, is moderate, balancing performance and computational demand well.

**Cuckoo Search:** Cuckoo Search stands out in Recall (0.496334) and F-measure (0.470321), suggesting it effectively identifies relevant items, even if its Precision is lower. However, its high time complexity $O(i \times p \times n \times m)$ may pose challenges for scalability, making it computationally intensive.

**Time-Based Decay:** With a simple time complexity of $O(n \times m)$, this method offers a good trade-off between speed and accuracy. While it doesn't lead in any specific metric, its balanced performance across all metrics makes it a reliable choice for large datasets.

**Decay Model Selection:** This method shows average performance across all metrics but benefits from a relatively simple time complexity of $O(f \times n \times m)$. It's suitable when resources are limited and performance needs are moderate.

**TCCF:** This method, with a time complexity of $O(n \times m \times j)$, generally shows lower performance compared to others, particularly in Recall and F-measure. It appears less effective in finding relevant recommendations.

**Recommended Methods:** Based on the analysis, **Content-Based Similarity** is recommended for its high accuracy and reasonable computational demands, making it ideal for scenarios where performance is critical. **Time-Based Decay** is also a strong candidate due to its simplicity and balanced metrics, suitable for applications where efficiency is a priority. These two methods provide the best combination of effectiveness and efficiency given the dataset and computational constraints.

### 5.2.2 Analysis of Methods in CellPhone & Accessories dataset

Below, Table 5-3 presents the evaluation metrics of various methods applied to the Cell-Phone dataset. For visual comparison, a histogram is also provided in the corresponding Figure 5-5.

***Table 5-3*** *Metrics value in CellPhone dataset*

| Metric | TCCF | Content-Based Similarity | Time-Based Decay | Cuckoo-Search | Decay-Model Selection |
|---|---|---|---|---|---|
| **MAE** | 1.240229 | 1.174422 | 1.125573 | 1.092541 | 1.222028 |
| **Precision** | 0.747393 | 0.785965 | 0.776438 | 0.850229 | 0.756755 |
| **Recall** | 0.900987 | 0.914825 | 0.906035 | 0.901812 | 0.906695 |
| **F-measure** | 0.670846 | 0.718873 | 0.698814 | 0.769483 | 0.681130 |

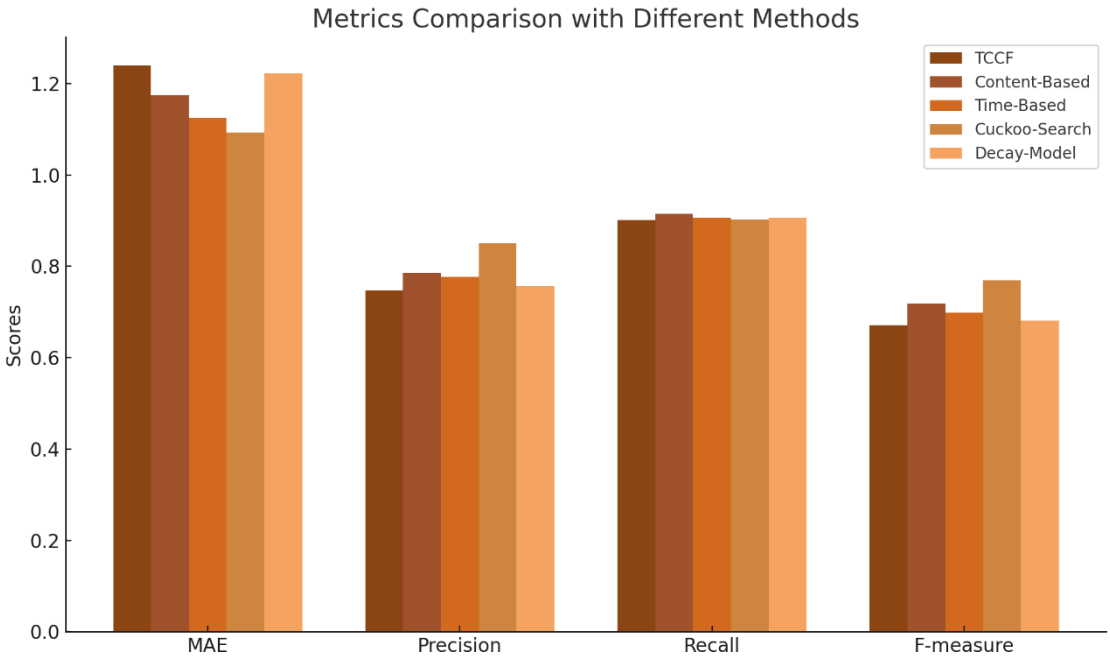


***Figure 5-5*** *Metrics comparison in CellPhone dataset*

**Content-Based Similarity:** This approach yields good recall (0.914825) and precision (0.785965) scores, suggesting useful suggestions. Because of its $O(n \times m \times d)$ time complexity, which strikes a fair balance between computational demand and performance, it can be used in a variety of applications.

**Time-Based Decay:** The technique has a low MAE (1.125573) and a simple temporal complexity of $O(n \times m)$. It also delivers great accuracy. While its precision and recall aren't the best, its balanced metrics make it a solid option for large datasets when efficiency is crucial.

**Cuckoo Search:** Cuckoo Search performs exceptionally well overall, as evidenced by its excellent precision (0.850229) and F-measure (0.769483). Its $O(i \times p \times n \times m)$ temporal complexity may pose a problem to scalability, though, as it is best suited for situations in which precision is more important than computing cost.

**Decay Model Selection:** This approach prioritizes recall and displays average performance across measures (0.906695). Because of its comparatively simple time complexity, $O(f \times n \times m)$, it is a good choice for situations where resources are limited and modest performance is sufficient.

**TCCF:** This approach performs usually worse, especially in the F-measure (0.670846), and has the greatest MAE (1.240229). Having an O(n×m×j) temporal complexity, it seems to be less efficient in locating pertinent recommendations.

**Recommended Methods:** The analysis indicates that **Cuckoo Search** is the best option when performance is critical due to its excellent precision and F-measure. Because of its balanced metrics and simplicity, **Time-Based Decay** is also a good option for applications that prioritize efficiency. The optimal balance between efficiency and computational viability is provided by these two techniques.

### 5.2.3 Analysis of Methods in VideoGames dataset

A comprehensive analysis of the performance indicators for various approaches when implemented on the videoGames dataset is presented in Table 5-4. Furthermore, Figure 5-6 provides a graphical depiction of these measurements using a histogram, facilitating the comparison of the outcomes among the several approaches.

***Table 5-4*** *Metrics value in VideoGames dataset*

| Metric | TCCF | Content-Based Similarity | Time-Based Decay | Cuckoo Search | Decay Model Selection |
|---|---|---|---|---|---|
| **MAE** | 1.347203 | 1.296863 | 1.316134 | 1.295237 | 1.366656 |
| **Precision** | 0.794163 | 0.828331 | 0.801661 | 0.852541 | 0.801641 |
| **Recall** | 0.803864 | 0.829198 | 0.813306 | 0.822620 | 0.809693 |
| **F-measure** | 0.627770 | 0.687088 | 0.643442 | 0.698377 | 0.642635 |

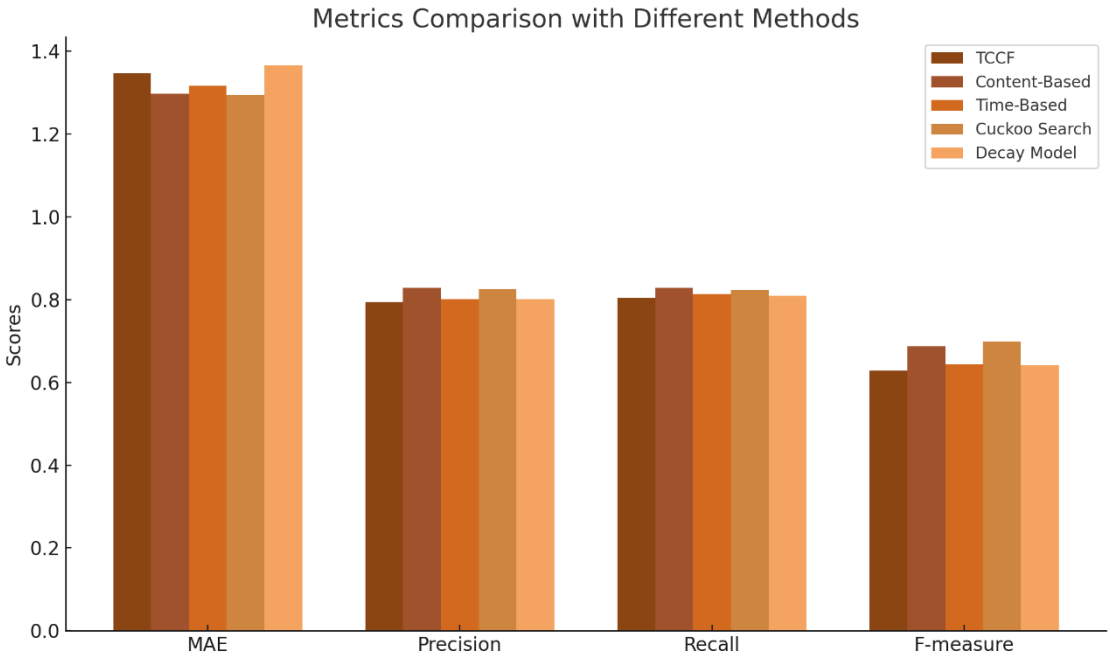


***Figure 5-6*** *Metrics comparison in VideoGames dataset*

**The Content-Based Similarity:** this approach demonstrates robust performance with a low Mean Absolute Error (MAE) of 1.296863 and a high precision of 0.828331, suggesting precise and pertinent recommendations. In addition, it maintains a reliable recall rate of 0.829198 and an F-measure of 0.687088. The temporal complexity of the system, $O(n \times m \times d)$, strikes a compromise between computing demand and efficiency.

**Time-Based Decay:** This merhod provides a satisfactory level of accuracy with a Mean Absolute Error (MAE) of 1.316134. The precision and recall, at 0.801661 and 0.813306 respectively, yield a balanced F-measure of 0.643442. Given its low temporal complexity of $O(n \times m)$, this method is highly efficient for handling larger datasets.

**Cuckoo Search:** the Cuckoo Search method demonstrates exceptional precision (0.852541) and F-measure (0.698377), indicating a highly effective overall performance. Achieving a low MAE of 1.295237 suggests a good level of accuracy. Nevertheless, the time complexity $O(i \times p \times n \times m)$ is greater, which may have an effect on scalability.

**Decay model selection:** this approach exhibits a superior Mean Absolute Error (MAE) of 1.366656 and tends to achieve average performance across many metrics, with a particular emphasis on recall of 0.809693. The complexity of the system, denoted as O(f×n×m), is moderately simple, which makes it appropriate for situations when computational resources are constrained.

**TCCF:** This approach exhibits the highest Mean Absolute Error (MAE) of 1.347203 and a lower F-measure of 0.627770, suggesting that its recommendations are less effective. The time complexity of the system, denoted as $O(n \times m \times j)$, is substantially high without any accompanying performance advantages.

**Recommended Methods:** Considering the analysis, **Cuckoo Search** is highly recommended because to its exceptional precision and F-measure, making it well-suited for situations where performance is of utmost importance. Furthermore, its balanced accuracy and computational economy make **Content-Based Similarity** a very promising option for a wide range of applications. These two approaches provide the optimal balance between efficacy and efficiency.

## 5.3 Determining Prominent Approaches to Recommender Systems

Based on various evaluation techniques, we can identify the top three methods among all the proposed and paper methods. The following explanation outlines the best three methods in the order presented below:

**Content-Based Similarity**

Content-Based Similarity stands out for its impressive accuracy and precision. It has demonstrated a low Mean Absolute Error (MAE), indicating its ability to provide highly accurate and relevant recommendations. This method is often highlighted for scenarios where performance is essential due to its balanced metrics. The computational complexity of Content-Based Similarity is moderate, denoted as O(n×m×d). This balance between computational demand and performance makes it applicable in a variety of contexts, where the available resources are sufficient for moderate complexity. Overall, Content-Based Similarity is ideal for applications requiring high accuracy and precision with reasonable computational constraints, offering a versatile solution across different scenarios.

**Cuckoo Search**

Cuckoo Search is renowned for its exceptional precision and F-measure, making it the method of choice for situations where performance is paramount. This method excels in identifying the most relevant items effectively, which is a significant advantage in precision-critical applications. However, the trade-off comes with its higher computational complexity, O(i×p×n×m), which may pose challenges for scalability and resource-intensive environments. Despite this, Cuckoo Search remains a top contender for applications where precision outweighs the need for computational efficiency. Its strong performance metrics make it particularly suitable when identifying precise recommendations is more important than processing speed.

**Time-Based Decay**

Time-Based Decay is celebrated for its simplicity and balanced performance across various metrics. It is an excellent choice for large datasets where efficiency is a priority. The method's low MAE in several instances suggests it maintains a good level of accuracy. With a simple time complexity of O(n×m), Time-Based Decay is highly efficient, making it well-suited for handling extensive datasets. This method is particularly appropriate for scenarios with limited computational resources, where its straightforward approach and balanced metrics provide a reliable and effective solution.

In conclusion, these three methods offer distinct advantages based on their strengths and application contexts. Content-Based Similarity provides a compelling balance of accuracy and computational efficiency, making it adaptable to various needs. Cuckoo Search excels in precision, suitable for scenarios where identifying the most relevant items is crucial. Meanwhile, Time-Based Decay delivers simplicity and efficiency, ideal for large-scale applications with resource constraints. Each method presents a unique combination of effectiveness and efficiency, tailored to different datasets and computational requirements.

## 5.4 Key observations derived from research

Assessment of leading methodologies for recommender systems offers valuable insights that might improve our comprehension and implementation of these technologies. Outlined below are essential points, structured to demonstrate a coherent development from approaches to user influence.

**1. Varied methodologies and their distinct advantages**

The analysis demonstrates that there is no typically superior approach for recommender systems. Instead, each methodology -- Content-Based Similarity, Cuckoo Search, and Time-Based Decay -- presents unique advantages that are suitable for diverse application scenarios. Content-Based Similarity is characterized by its exceptional accuracy and precision, making it well-suited for situations when peak performance is of utmost importance. However, Cuckoo Search achieves exceptional performance in applications that require precision, even though it has a greater computational complexity. The presence of much variation emphasizes the critical need of meticulous choice of methodologies according to the particular project needs, accessible resources, and intended results.

**2. Striking a balance between performance and computational efficiency**

The results underscore the need to attaining an appropriate balance between performance indicators and computational resource utilization. Although Cuckoo Search offers outstanding accuracy, its considerable computing requirements may impede scaling in settings with limited resources. By contrast, Time-Based Decay provides a more straightforward and effective approach for handling extensive datasets, making it especially appropriate for applications that have restricted computing capabilities. Achieving this equilibrium is crucial for the development of scalable and efficient recommender systems that can function effectively in practical situations.

### 3. Resolving constraints in Time-Aware Recommendation Systems

Although there have been notable improvements in time-aware recommendation algorithms, there are still substantial constraints, especially in their capacity to adjust to the changing characteristics of user preferences. The investigation reveals that current models, especially those that use the Time Correlation Coefficient (TCC), frequently have difficulties in accurately representing the changing changes in user interests. Hence, it is imperative to consistently improve these models by including temporal dynamics and item similarity, as these factors are crucial for producing precise and pertinent recommendations.

### 4. Novel Strategies for Improving Precision

The suggested improvements to the TCC model, including the incorporation of item similarity and investigation of dynamic attenuation coefficients, offer valuable opportunities for further study and advancement. The objective of these advancements is to enhance the precision of time-aware recommendation systems by more accurately representing the impact of time on user preferences. It is crucial to prioritize ongoing innovation and adaptation in order to effectively meet the evolving user behaviors and preferences, therefore ensuring the continued relevance and effectiveness of recommender systems.

### 5. Augmenting user experience and involvement

Fundamentally, the main objective of improving these recommender systems is to augment user experience and involvement. Through providing suggestions that closely correspond to the ever-changing nature of user preferences, organizations can promote higher levels of pleasure and engagement. Emphasizing user-centric design is crucial for the optimal performance of any recommender system, since it has a direct impact on user retention, loyalty, and overall happiness.

Ultimately, the results obtained from the assessment of suggested new approaches offer a very useful understanding of their advantages, constraints, and real-world utilities. The effectiveness and relevance of recommender systems in different domains can be greatly improved by comprehending the different strengths of each approach, achieving a balance between performance and computational efficiency, overcoming the constraints of time-aware systems, and prioritizing innovative solutions. The primary objective of this dedication to ongoing enhancement is to enhance user pleasure and involvement, thereby guaranteeing that recommender systems effectively cater to the changing requirements of users in a dynamic setting.

## 5.5 Limitations and Potential Challenges

While the study offers valuable insights into recommender systems, there are a few important limitations and challenges that could affect the applicability of the findings. Three key areas are:

1. **Dependence on Specific Datasets**:

   The study's conclusions are based on the performance of the methods tested on specific datasets (Amazon and MovieLens). Different datasets may exhibit varied user behaviors, item diversity, or interaction patterns, which could significantly affect the performance of the evaluated approaches. This limits the generalizability of the results, as methods that work well on these datasets may not perform as effectively on others with different characteristics.

2. **Scalability and Computational Complexity**:

   While the computational complexity of the methods is analyzed, the study doesn't fully address the practical implications of implementing them in real-world environments. For instance, the high complexity of Cuckoo Search, despite its strong precision, can limit its scalability in resource-constrained or large-scale settings. This could hinder its effectiveness in applications where real-time processing and efficiency are critical.

3. **External Validity and Practical Implications**:

   Finally, the practical application of these methods in real-world systems may encounter unforeseen challenges that are not captured in the experimental environment. For example, user feedback loops, recommendation diversity, and system latency are factors that can influence the success of a recommender system in practice. The absence of these real-world considerations in the evaluation presents a threat to the external validity of the results, meaning the effectiveness of these approaches may differ when deployed in live systems.

   In conclusion, while this study provides a strong foundation for understanding the strengths of various recommender system approaches, addressing these limitations and threats to validity is crucial for the accurate interpretation and application of the findings in broader contexts. Future work should consider further exploration of these limitations and seek to enhance the robustness and generalizability of the methods.

# 6. CONCLUSION

The exploration of time-aware recommendation systems within this thesis has yielded significant insights into the efficacy and limitations of current algorithms, particularly those employing the Time Correlation Coefficient (TCC) approach. While these models exhibit commendable effectiveness, their inability to fully capture the dynamic and nuanced nature of user preferences over time remains a challenge. This research has addressed these shortcomings by proposing enhancements that integrate temporal dynamics and item similarity into the recommendation process. This study introduced and evaluated four innovative methodologies: Content-based, Cuckoo Search, Decay Model, and Time-Based. Each approach offers unique strengths, underscoring the importance of selecting methodologies aligned with specific application requirements and resource constraints. Content-based provides high accuracy with moderate computational demands, making it suitable for balanced performance needs. Cuckoo Search excels in precision, ideal for environments where accuracy is paramount despite higher computational costs. Time-based offers simplicity and efficiency, perfect for large-scale applications with limited resources. Key observations from this research highlight the necessity of balancing performance with computational efficiency, recognizing that no single approach universally outperforms others. Instead, each methodology offers distinct advantages suited to varying contexts and needs. The enhancements proposed for the TCC model, present promising avenues for further refinement of time-aware recommendation systems. Ultimately, the advancements delineated in this thesis aim to elevate user experience and engagement by delivering recommendations that adapt closely to the evolving preferences of users. By emphasizing user-centric design and continuous innovation, organizations can achieve higher levels of satisfaction, retention, and loyalty.

While this research has made significant strides in improving time-aware recommendation systems, there remain several areas for future exploration. One key direction is to further investigate the generalizability of the proposed methods across diverse datasets with varying characteristics. Extending the evaluation to a broader range of datasets, including those with different user behavior patterns, item types, and temporal trends, will help refine the algorithms and ensure their robustness in different domains. In addition, incorporating real-world user feedback loops will be essential for enhancing the practical utility of the proposed approaches. By focusing on these areas, future research can help bridge the gap between experimental results and real-world applications, ensuring that recommendation systems continue to evolve and meet the dynamic needs of users.

# REFERENCES


Adomavicius, G., and A. Tuzhilin. 2005a. “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(6):734–49. doi: 10.1109/TKDE.2005.99.

Adomavicius, G., and A. Tuzhilin. 2005b. “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(6):734–49. doi: 10.1109/TKDE.2005.99.

Adomavicius, Gediminas, and Jingjing Zhang. 2012. “Impact of Data Characteristics on Recommender Systems Performance.” *ACM Trans. Manage. Inf. Syst.* 3(1). doi: 10.1145/2151163.2151166.

Ahmadian, Sajad, Nima Joorabloo, Mahdi Jalili, and Milad Ahmadian. 2022. “Alleviating Data Sparsity Problem in Time-Aware Recommender Systems Using a Reliable Rating Profile Enrichment Approach.” *Expert Systems with Applications* 187:115849. doi: https://doi.org/10.1016/j.eswa.2021.115849.

Ahmed, Ismail, Al-Qasem Al-Hadi, Nurfadhlina Bt, Mohd Sharef, Nurfadhlina Mohd, Sharef Md, Nasir Sulaiman, and Norwati Mustapha. 2017. *Review of the Temporal Recommendation System with Matrix Factorization*. Vol. 13.

Alabduljabbar, R, M. Alshareef, and N. Alshareef. 2023. “Time-Aware Recommender Systems: A Comprehensive Survey and Quantitative Assessment of Literature.” *IEEE Access* 11:45586–604. doi: 10.1109/ACCESS.2023.3274117.

Alabduljabbar, Reham, Manal Alshareef, and Nada Alshareef. 2023. “Time-Aware Recommender Systems: A Comprehensive Survey and Quantitative Assessment of Literature.” *IEEE Access* 11:45586–604. doi: 10.1109/ACCESS.2023.3274117.

Bao, Jianghan, and Yu Zhang. 2021. “Time-Aware Recommender System via Continuous-Time Modeling.” Pp. 2872–76 in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, *CIKM ’21*. New York, NY, USA: Association for Computing Machinery.

Bartocci, Ezio, Ebru Aydin Gol, Iman Haghighi, and Calin Belta. 2018. “A Formal Methods Approach to Pattern Recognition and Synthesis in Reaction Diffusion Networks.” *IEEE Transactions on Control of Network Systems* 5(1):308–20. doi: 10.1109/TCNS.2016.2609138.

Bobadilla, J., F. Ortega, A. Hernando, and A. Gutiérrez. 2013a. “Recommender Systems Survey.” *Knowledge-Based Systems* 46:109–32. doi: https://doi.org/10.1016/j.knosys.2013.03.012.

Bobadilla, J., F. Ortega, A. Hernando, and A. Gutiérrez. 2013b. “Recommender Systems Survey.” *Knowledge-Based Systems* 46:109–32. doi: https://doi.org/10.1016/j.knosys.2013.03.012.

Breese, John S., David Heckerman, and Carl Kadie. n.d. *Empirical Analysis of Predictive Algorithms for Collaborative Filtering*.

Bu, Jiajun, Xin Shen, Bin Xu, Chun Chen, Xiaofei He, and Deng Cai. 2016. “Improving Collaborative Recommendation via User-Item Subgroups.” *IEEE Transactions on Knowledge and Data Engineering* 28(9):2363–75. doi: 10.1109/TKDE.2016.2566622.

Burke, Robin. 2002. “Hybrid Recommender Systems: Survey and Experiments.” *User Modeling and User-Adapted Interaction* 12(4):331–70. doi: 10.1023/A:1021240730564.

Cai, Ruichu, Ruming Lu, Wei Chen, and Zhifeng Hao. 2024. “Counterfactual Contextual Bandit for Recommendation under Delayed Feedback.” *Neural Computing and Applications* 36(23):14599–613. doi: 10.1007/s00521-024-09800-0.

Chen, Anne Yun-An, and Dennis McLeod. 2006. “Collaborative Filtering for Information Recommendation Systems.” Pp. 118–23 in *Encyclopedia of E-Commerce, E-Government, and Mobile Commerce*. IGI Global.

Chen, Chen, Hongzhi Yin, Junjie Yao, and Bin Cui. 2013. “TeRec: A Temporal Recommender System over Tweet Stream.” *Proc. VLDB Endow.* 6(12):1254–57. doi: 10.14778/2536274.2536289.

Chen, Long-Sheng, Fei-Hao Hsu, Mu-Chen Chen, and Yuan-Chia Hsu. 2008. “Developing Recommender Systems with the Consideration of Product Profitability for Sellers.” *Information Sciences* 178(4):1032–48. doi: https://doi.org/10.1016/j.ins.2007.09.027.

Chua, Freddy Chong Tat, Richard J. Oentaryo, and Ee-Peng Lim. 2013. “Modeling Temporal Adoptions Using Dynamic Matrix Factorization.” Pp. 91–100 in *2013 IEEE 13th International Conference on Data Mining*.

Cui, Zhihua, Xianghua Xu, Fei Xue, Xingjuan Cai, Yang Cao, Wensheng Zhang, and Jinjun Chen. 2020. “Personalized Recommendation System Based on Collaborative Filtering for IoT Scenarios.” *IEEE Transactions on Services Computing* 13(4):685–95. doi: 10.1109/TSC.2020.2964552.

Ding, Yi, and Xue Li. 2005. “Time Weight Collaborative Filtering.” Pp. 485–92 in *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, *CIKM ’05*. New York, NY, USA: Association for Computing Machinery.

Friedman, Nir, Dan Geiger, and Moises Goldszmidt. 1997. “Bayesian Network Classifiers.” *Machine Learning* 29(2):131–63. doi: 10.1023/A:1007465528199.

Hamzehei, Asso, Raymond K. Wong, Danai Koutra, and Fang Chen. 2019. “Collaborative Topic Regression for Predicting Topic-Based Social Influence.” *Machine Learning* 108(10):1831–50. doi: 10.1007/s10994-018-05776-w.

Harper, F. M. ,. &. Konstan, J. A. 2015. “The MovieLens Datasets: History and Context.” *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5(9):1–19.

Herlocker, Jonathan L., Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. 2004. “Evaluating Collaborative Filtering Recommender Systems.” *ACM Trans. Inf. Syst.* 22(1):5–53. doi: 10.1145/963770.963772.

Hou, Yupeng, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. 2024. “Bridging Language and Items for Retrieval and Recommendation.”

Isinkaye, F. O., Y. O. Folajimi, and B. A. Ojokoh. 2015. “Recommendation Systems: Principles, Methods and Evaluation.” *Egyptian Informatics Journal* 16(3):261–73.

Jannach, Dietmar and Zanker, Markus and Felfernig, Alexander and Friedrich, Gerhard. 2010. *Recommender Systems: An Introduction*. Cambridge University Press.

Koren, Yehuda. 2009. “Collaborative Filtering with Temporal Dynamics.” Pp. 447–56 in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *KDD ’09*. New York, NY, USA: Association for Computing Machinery.

Koren, Yehuda, and Robert Bell. 2015. “Advances in Collaborative Filtering.” Pp. 77–118 in *Recommender Systems Handbook*, edited by F. Ricci, L. Rokach, and B. Shapira. Boston, MA: Springer US.

Li, Taoying, Linlin Jin, Zebin Wu, and Yan Chen. 2019. “Combined Recommendation Algorithm Based on Improved Similarity and Forgetting Curve.” *Information (Switzerland)* 10(4). doi: 10.3390/info10040130.

Lops Pasquale and de Gemmis, Marco and Semeraro Giovanni. 2011. “Content-Based Recommender Systems: State of the Art and Trends.” Pp. 73–105 in *Recommender Systems Handbook*, edited by L. and S. B. and K. P. B. Ricci Francesco and Rokach. Boston, MA: Springer US.

Luo, Xin, Yunni Xia, and Qingsheng Zhu. 2013. “Applying the Learning Rate Adaptation to the Matrix Factorization Based Collaborative Filtering.” *Knowledge-Based Systems* 37:154–64. doi: https://doi.org/10.1016/j.knosys.2012.07.016.

Nicolas Hug. 2017. “Surprise: A Python Library for Recommender Systems.”

OpenAI. 2024. “GPT-4 Model.” *OpenAI*.

Pang, Shanchen, Shihang Yu, Guiling Li, Sibo Qiao, and Min Wang. n.d. “TIME-SENSITIVE COLLABORATIVE FILTERING ALGORITHM WITH FEATURE STABILITY.” *Computing and Informatics* 39(2020):141–55. doi: 10.31577/cai.

Resnick, Paul, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. “GroupLens: An Open Architecture for Collaborative Filtering of Netnews.” Pp. 175–86 in *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, *CSCW ’94*. New York, NY, USA: Association for Computing Machinery.

Rostami, Mehrdad, Vahid Farrahi, Sajad Ahmadian, Seyed Mohammad Jafar Jalali, and Mourad Oussalah. 2023. “A Novel Healthy and Time-Aware Food Recommender System Using Attributed Community Detection.” *Expert Systems with Applications* 221:119719. doi: https://doi.org/10.1016/j.eswa.2023.119719.

Salah, Aghiles, Nicoleta Rogovschi, and Mohamed Nadif. 2016. “A Dynamic Collaborative Filtering System via a Weighted Clustering Approach.” *Neurocomputing* 175:206–15. doi: https://doi.org/10.1016/j.neucom.2015.10.050.

Sarwar, Badrul, George Karypis, Joseph Konstan, and John Riedl. 2001. “Item-Based Collaborative Filtering Recommendation Algorithms.” Pp. 285–95 in *Proceedings of the 10th International Conference on World Wide Web*, *WWW ’01*. New York, NY, USA: Association for Computing Machinery.

Sayyed, F. R., R. V Argiddi, and S. S. Apte. 2013. *GENERATING RECOMMENDATIONS FOR STOCK MARKET USING COLLABORATIVE FILTERING*. Vol. 46.

Schafer J. Ben and Frankowski, Dan and Herlocker Jon and Sen Shilad. 2007. “Collaborative Filtering Recommender Systems.” Pp. 291–324 in *The Adaptive Web: Methods and Strategies of Web Personalization*, edited by A. and N. W. Brusilovsky Peter and Kobsa. Berlin, Heidelberg: Springer Berlin Heidelberg.

Shi, Yue, Martha Larson, and Alan Hanjalic. 2014. “Collaborative Filtering beyond the User-Item Matrix: A Survey of the State of the Art and Future Challenges.” *ACM Comput. Surv.* 47(1). doi: 10.1145/2556270.

Stern, David H., Ralf Herbrich, and Thore Graepel. 2009. “Matchbox: Large Scale Online Bayesian Recommendations.” Pp. 111–20 in *Proceedings of the 18th International Conference on World Wide Web*, *WWW ’09*. New York, NY, USA: Association for Computing Machinery.

Su, Hongyi, Xianfei Lin, Bo Yan, and Hong Zheng. 2015. “The Collaborative Filtering Algorithm with Time Weight Based on MapReduce.” Pp. 386–95 in *Big Data Computing and Communications*, edited by Y. Wang, H. Xiong, S. Argamon, X. Li, and J. Li. Cham: Springer International Publishing.

Sun, Baoshan, and Lingyu Dong. 2017. “Dynamic Model Adaptive to User Interest Drift Based on Cluster and Nearest Neighbors.” *IEEE Access* 5:1682–91. doi: 10.1109/ACCESS.2017.2669243.

Wan, Yongquan, Yizhou Chen, and Cairong Yan. 2021. “An Integrated Time-Aware Collaborative Filtering Algorithm.” Pp. 369–79 in *Knowledge Management in Organizations*, edited by L. Uden, I.-H. Ting, and K. Wang. Cham: Springer International Publishing.

Wang, Hao, Naiyan Wang, and Dit-Yan Yeung. 2015. “Collaborative Deep Learning for Recommender Systems.” Pp. 1235–44 in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *KDD ’15*. New York, NY, USA: Association for Computing Machinery.

Wangwatcharakul, C., and S. Wongthanavasu. 2020. “Dynamic Collaborative Filtering Based on User Preference Drift and Topic Evolution.” *IEEE Access* 8:86433–47. doi: 10.1109/ACCESS.2020.2993289.

Xiang, Liang, Quan Yuan, Shiwan Zhao, Li Chen, Xiatian Zhang, Qing Yang, and Jimeng Sun. 2010. “Temporal Recommendation on Graphs via Long- and Short-Term Preference Fusion.” Pp. 723–32 in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *KDD ’10*. New York, NY, USA: Association for Computing Machinery.

Xiong, Liang, Xi Chen, Tzu-Kuo Huang, Jeff Schneider, and Jaime G. Carbonell. n.d. “Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorization.” Pp. 211–22 in *Proceedings of the 2010 SIAM International Conference on Data Mining (SDM)*.

Xue, Taofeng, Beihong Jin, Beibei Li, Weiqing Wang, Qi Zhang, and Sihua Tian. 2019a. “A Spatio-Temporal Recommender System for On-Demand Cinemas.” Pp. 1553–62 in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, *CIKM ’19*. New York, NY, USA: Association for Computing Machinery.

Xue, Taofeng, Beihong Jin, Beibei Li, Weiqing Wang, Qi Zhang, and Sihua Tian. 2019b. “A Spatio-Temporal Recommender System for On-Demand Cinemas.” Pp. 1553–62 in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, *CIKM ’19*. New York, NY, USA: Association for Computing Machinery.

Zahra, Sobia, Mustansar Ali Ghazanfar, Asra Khalid, Muhammad Awais Azam, Usman Naeem, and Adam Prugel-Bennett. 2015. “Novel Centroid Selection Approaches for KMeans-Clustering Based Recommender Systems.” *Information Sciences* 320:156–89. doi: https://doi.org/10.1016/j.ins.2015.03.062.

Zhang, Wei, and Jianyong Wang. 2016. “Integrating Topic and Latent Factors for Scalable Personalized Review-Based Rating Prediction.” *IEEE Transactions on Knowledge and Data Engineering* 28(11):3013–27. doi: 10.1109/TKDE.2016.2598740.

Zhao, Zhi-Dan, and Ming-sheng Shang. 2010. “User-Based Collaborative-Filtering Recommendation Algorithms on Hadoop.” Pp. 478–81 in *2010 Third International Conference on Knowledge Discovery and Data Mining*.

Zhou, Zhangbing, Zehui Cheng, Liang-Jie Zhang, Walid Gaaloul, and Ke Ning. 2018. “Scientific Workflow Clustering and Recommendation Leveraging Layer Hierarchical Analysis.” *IEEE Transactions on Services Computing* 11(1):169–83. doi: 10.1109/TSC.2016.2542805.

Zhu, XiaoMing, HongWu Ye, and SongJie Gong. 2009. “A Personalized Recommendation System Combining Case-Based Reasoning and User-Based Collaborative Filtering.” Pp. 4026–28 in *2009 Chinese Control and Decision Conference*.

Ziegler, Cai-Nicolas, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. 2005. “Improving Recommendation Lists through Topic Diversification.” Pp. 22–32 in *Proceedings of the 14th*

*International Conference on World Wide Web*, *WWW '05*. New York, NY, USA: Association for Computing Machinery.