Received 25 November 2024, accepted 18 December 2024, date of publication 24 December 2024, date of current version 31 December 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3522118

RESEARCH ARTICLE

From Code to Ratings: Converting Programming Data to Enhance Collaborative Filtering in Educational Online Judge Systems

DANIEL M. MUEPU[1,2], **(Graduate Student Member, IEEE),**
AND YUTAKA WATANOBE[1], **(Member, IEEE)**
[1]Graduate Department of Computer and Information System, The University of Aizu, Aizuwakamatsu, Fukushima 965-8580, Japan
[2]Department of Computing, Management and English for Business, University of Kinshasa, Kinshasa, Democratic Republic of the Congo

Corresponding author: Daniel M. Muepu (mdaniel108@gmail.com)

This work was supported by the Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant 23H03508.

ABSTRACT This study introduces and compares three innovative approaches for recommending programming problems within an Online Judge system (OJ), tackling the challenge of deriving implicit ratings from user interactions without explicit user ratings. Conventional collaborative filtering (CF) methods often struggle with the sparse and implicit feedback typical in educational settings, limiting recommendation accuracy. A common method to handle implicit feedback is to use a binary system, which in the context of programming problems, simply records whether a user solved a problem (1) or did not (0). However, this method lacks scalability, fails to capture satisfaction levels, and overlooks problem difficulty and user skill. We analyzed submissions on the OJ to develop methods that generate implicit ratings reflecting user engagement and problem-solving performance. We proposed three distinct approaches, i.e., a Direct Weighted Sum Approach, a Formula-Based Approach and a Distance-Based Approach. Each approach determines user satisfaction, scaled from 1 to 5, and utilizes this in a Neural CF model to improve recommendation accuracy. Additionally, we included a baseline dataset that categorizes exercises in a binary format, marking them as either solved or not. The results demonstrate that all the approaches performed well in terms of accuracy. However, the proposed approaches significantly outperformed the baseline approach in recommendation quality by presenting relevant items to users earlier and offering a broader range of items. Another advantage of the proposed approaches is their consideration of problem difficulty and user skill, enabling a better understanding of how challenging each recommended item might be for each user.

INDEX TERMS Recommender systems, neural collaborative filtering, programming education, online judge system, implicit rating.

I. INTRODUCTION

The advent of the internet and digital technologies has revolutionized the educational landscape, fundamentally transforming how knowledge is disseminated and acquired [1], [2]. The shift towards online education platforms has democratized access to learning, transcending geographical barriers and making a wide range of educational content accessible to a global audience [3]. This digital transformation has been particularly pivotal in specialized fields such as programming, where the rapid evolution of technology necessitates continual learning and skill development [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Pasquale De Meo.

Programming, a critical skill in the digital economy, requires a robust learning framework due to its complex, abstract concepts and practical application [5], [6], [7]. Traditional classroom-based approaches often struggle to keep pace with the dynamic nature of the field, leading to a growing reliance on online platforms for programming education [8]. These platforms offer an array of resources – from interactive tutorials to coding exercises and real-world problem-solving scenarios. Online Judge Systems (OJ) have emerged as popular platforms in programming education, providing a structured environment for practicing coding skills [9]. OJ present learners with a vast array of problems of varying difficulty levels, allowing them to practice, experiment, and receive immediate feedback [10], [11].

However, the abundance of resources on OJ platforms brings its own challenges, including the overwhelming choice for learners and the need for guided, personalized learning pathways [12], [13], [14]. While OJs offer a practical avenue for skill development, navigating the vast array of problems and identifying the most appropriate exercises can be daunting for learners [15], [16].

Initially developed for the e-commerce sector to enhance customer experience by recommending products, recommendation systems (RS) have found new applications in diverse domains, including education. While RSs have their roots in business applications, their adaptation in educational contexts introduces unique challenges. The educational domain demands an integration of pedagogical principles, domain ontology, and an understanding of diverse learning processes [17]. Collaborative filtering (CF), a fundamental technique in RSs, operates on the principle of identifying patterns in user behavior to make recommendations. CF is based on the assumption that users who agreed in the past will agree in the future [18].

The main problem when using CF in programming education is the lack of explicit ratings for programming problems, which are a core element of CF [19], [20]. In typical CF systems, users rate their preferences for items on a scale, allowing the system to identify similarities in ratings and make recommendations [21]. However, programming problems are not usually rated by learners. Additionally, the objective in a learning context is not merely to engage with content that aligns with existing preferences but to encourage learners to tackle a comprehensive range of problems. This situation necessitates a RS capable of dynamically adjusting to a learner's progress, continuously adapting to their strengths and weaknesses [22].

To adapt programming data for a Collaborative Filtering (CF) model, one approach is to simplify user interactions into a binary format, classifying exercises as either completed (1) or uncompleted (0). While this binary model is straightforward, it fails to capture the more nuanced aspects of user experience. Additionally, it does not account for problem difficulty and user skill level, which can result in recommending very easy problems to experienced users, leading to boredom. A user might solve an exercise (indicated as 1), but the process might have been extremely challenging or frustrating, which a binary value cannot convey. Similarly, an exercise marked as 0 (unsolved) does not differentiate between a user who almost solved it and one who barely understood it. In contrast, traditional CF systems rely on explicit ratings—such as a 5/5 or 3/5 score—that directly reflect a user's satisfaction or preference, offering a richer understanding of user engagement.

To address the challenge of effectively guiding learners through the vast range of programming problems, we propose three approaches that adapt traditional CF techniques to the unique needs of programming education. Since traditional CF methods rely on explicit user ratings, which are often absent in educational contexts, our approaches introduce an innovative way to interpret and leverage submission logs as a proxy. These approaches are based on the understanding that a learner's interaction with a programming problem—through attempts, errors, and eventual success or failure—provides valuable insights into users learning experience and satisfaction. Analyzing these submission logs allows us to generate implicit scaled rating values that reflect the learner's engagement and achievement, transforming the data into a form that traditional CF algorithms can utilize.

Having made the data suitable for CF, we then leveraged Neural Collaborative Filtering (NCF) to enhance the recommendation process. NCF, which combines the strengths of traditional CF with the advanced capabilities of neural networks, offers a more nuanced and effective approach to understanding and predicting user preferences [23]. In our context, NCF is adept at deciphering the complex, non-linear relationships inherent in the programming problem-solving data, enabling it to make highly personalized and pedagogically sound problem recommendations [24].This results in a RS that is not only more aligned with the individual learner's current capabilities and learning trajectory but also capable of adapting to their evolving skill levels and preferences.

The key contributions of this paper are threefold:

- First, we proposed three innovative approaches to simulate explicit ratings using data submitted by users, which were then applied to Neural Collaborative Filtering. These ratings express different levels of user satisfaction when solving a problem.
- Second, our proposed approaches enable the calculation of the difficulty level of each problem and the skill level of each user. This dual classification allows us to estimate user performance for every recommended problem, enhancing the personalization and effectiveness of the recommendations.
- Third, our proposed approaches significantly outperform the baseline approach, particularly in delivering pertinent and a wide range of content, demonstrating their superior ability to provide users with valuable and diverse recommendations.

The remainder of this paper is organized as follows: Section II presents an overview of the relevant background and prior studies in the field. Section III details the methodology used to develop our proposed approaches. Section IV describes the experimental setup, while Section V discusses the results. Section VI provides an in-depth discussion of the each approach's performance and its implications. Finally, Section VII concludes the paper by summarizing the key findings, highlighting contributions, and suggesting directions for future research.

II. BACKGROUND AND RELATED WORK

A. RECOMMENDATION SYSTEMS

RSs have gained significant prominence in the past few decades, particularly with the rise of online services such as YouTube and Amazon [25]. These systems use algorithms to filter information and provide personalized recommendations to users based on their past behavior or preferences [26], [27]. Their applications span across various industries, from e-commerce, where they suggest relevant products to buyers, to online advertising, where they match user preferences with relevant content. The efficiency of RSs can be critical to the success of some industries, as they can generate substantial income and provide a competitive edge over competitors [28], [29].

The field of RSs is broad and multifaceted, comprising several distinct approaches, each with its unique methodology and application spectrum [30]. Among the prominent techniques in RSs, CF stands out for its user-centric approach. CF operates on the fundamental premise that users who shared similar preferences in the past are likely to continue doing so in the future [31]. This technique involves analyzing the behaviors and preferences of a group of users and then making recommendations based on this collective information [32]. The underlying assumption is that if a user A has the same opinion as a user B on an issue, A is more likely to have B's opinion on a different issue than that of a randomly chosen user. This method has seen widespread adoption due to its simplicity and effectiveness, especially in domains like e-commerce, online streaming, and social media platforms.

Content-Based Filtering, on the other hand, takes a more item-centric approach. This method focuses on the attributes or features of the items themselves to make recommendations [33]. For instance, in a movie RS, attributes like genre, director, or cast form the basis for suggesting similar movies to users. The key advantage of this approach is its ability to provide highly tailored suggestions based on the user's unique preferences [34]. However, it often lacks the ability to introduce users to entirely new genres or styles, which they might not have explored previously.

Knowledge-Based systems represent another dimension of RSs. They operate on explicit inputs or queries from the user [35]. These systems are particularly useful in scenarios where user preferences are not well-defined or when there is a lack of historical data. For instance, in recommending travel packages or financial products, where user needs are specific and diverse, knowledge-based recommendations can be highly effective.

The Hybrid recommendation technique, as the name suggests, combines elements from different approaches, often resulting in a more sophisticated and well-rounded recommendation engine [36]. For example, a hybrid system might blend Collaborative and Content-Based Filtering to compensate for the limitations of each. This approach can provide more comprehensive recommendations by leveraging both the collective intelligence of user behavior and the specific attributes of the items.

CF system is primarily divided into two main types: User-Based and Item-Based CF [37]. User-Based CF (Figure 1) focuses on finding users who are similar to the target user and then suggesting items that these similar users have liked [38]. This method assumes that similar users will have similar tastes. For example, if user A and user B both liked items 1 and 2, and user B also likes item 3, then the system would recommend item 3 to user A. Item-Based CF (Figure 2) takes a different approach by focusing on the similarity between items. For example, if user D likes item 1, the system will look at the preferences of other users who also liked item 1 [39]. The system then identifies other items that these users commonly like and recommends those items to user D. The idea is that items liked by similar users are likely to be of interest to user D as well.

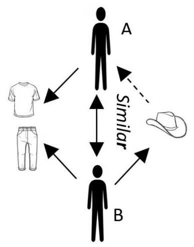

FIGURE 1. User-based CF.

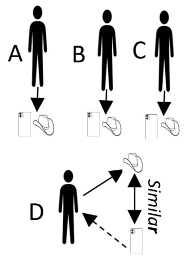

FIGURE 2. Item-based CF.

B. EDUCATIONAL RECOMMENDATION SYSTEMS

In recent years, RSs have also gained traction in the field of education. With the advent of online learning platforms, there has been an increasing need to personalize the learning experience for each student. Educational RSs analyze data from student performance, behavior, and interactions with the platform to make suggestions that optimize learning outcomes [40]. Educational RSs offer the potential to improve student engagement, retention, and accomplishment by personalizing the learning experience to each student. These recommendations can be based on different techniques, including CF, content-based filtering, or hybrid methods, and are designed to provide students with a seamless and enjoyable experience while boosting engagement and satisfaction [41], [42].

In [43], The authors conducted a systematic review on 98 articles to examine the existing research on RSs in support of educational practices. Their objective was to gather information on the educational contexts and subjects covered, the approaches used for development, and the recommended features. They also aimed to identify any gaps in the research that could inform future work. They found that collaborative, content-based, and hybrid approaches were the most commonly used techniques in RSs. In [44], The authors conducted a literature review to analyze recommender systems implemented in *MOOC*. The objective was to identify trends and insights reported in the academic literature on RSs in MOOCs. The review covers different types of recommendation techniques, recommendation types, and evaluation techniques used and reported in 67 papers. The authors designed a framework that classifies literature based on the design and evaluation aspects of RS in MOOCs. Finally, the review identifies gaps in the literature on this topic.

It is possible to recommend different learning materials for courses. In [45], the authors propose an online education course recommendation algorithm based on path factors to find and recommend similar courses more reasonably. For exercises, in [46], authors analyze learners' historical log data in a computer network course to find the similarities between a learner's knowledge and the required knowledge to solve exercises and make recommendations.

C. PROGRAMMING PROBLEMS RECOMMENDER SYSTEMS

Various approaches have been employed to support learners in selecting programming exercises. Some researchers have utilized content-based filtering, such as in [47], where a method was proposed to recommend programming problems with similar content and source code to a target assignment.

Other researchers have adapted collaborative filtering (CF) for the context of online judges (OJs). For example, in [48], a three-level CF recommendation method was developed to suggest alternative problems that users might find interesting. Similarly, [49] introduced an approach using an enriched user-problem matrix to enhance student representation, enabling more accurate neighborhood computations and recommendations. This method also incorporated a preprocessing step to address anomalous user behavior that could affect the recommendation process. In [50], the authors proposed a recommendation framework combining collaborative filtering and fuzzy logic to manage uncertainty in problem selection. This approach used collaborative filtering to model user preferences while leveraging fuzzy logic to handle uncertainties in the recommendation process. Real-world data evaluations demonstrated that this framework outperformed traditional methods, particularly for short recommendation lists.

This paper stands apart from previous studies by introducing new approaches to applying collaborative filtering. These innovative methods begin with calculating problem difficulty, estimating user skill levels, and evaluating user performance when solving programming problems. The next step integrates these calculated features to estimate user satisfaction on a scaled basis. The generated user satisfaction scores are then utilized in a collaborative filtering model, enhancing the recommendation process by incorporating these nuanced metrics.

III. METHODOLOGY

A. BASELINE APPROACH OVERVIEW

Table 1 shows an example of a submission log from the OJ. The User and Problem columns identify each user and their attempted problems. The Verdict column indicates if the user's solution was correct (Accepted) or not, and the Date column shows the submission date in Linux timestamp format.

TABLE 1. Example of a submission log.

user	problem	verdict	date
1	A	Accepted	1403864355640
2	A	Compile Error	1403864348000
2	A	Wrong Answer	1403864349500
2	A	Accepted	1403864351000
2	A	Wrong Answer	1403864352000
3	A	Accepted	1403864380420
2	B	Accepted	1403864353000
4	B	Compile Error	1403864348500
5	B	Wrong Answer	1403864349500
5	B	Wrong Answer	1403864351000
5	B	Compile Error	1403864352500

Table 2 shows how the baseline approach changes the User-Problem data in a binary way, where 1 indicates that the final verdict is "Accepted" and 0 indicates it is not. Let's consider Users 1, 2, and 3 as examples. Since all three solved Problem A, the Verdict is recorded as 1, meaning they are seen as similar. However, if we look at the original submission log in Table 1, there are important differences: Users 1 and 3 solved the problem on their first try, while User 2 needed four attempts. This distinction is not reflected in the transformed dataset, where all users are treated as equally proficient when solving Problem A.

TABLE 2. Baseline approach dataset.

user	problem	verdict
1	A	1
2	A	1
3	A	1
2	B	1
4	B	0
5	B	0

Moreover, this approach does not consider changes in user skill over time. For instance, User 1 might have solved the problem a long time ago his skill level may have improved since then. If User 3 is novice, recommending problems to User 1 based on User 3's activity could lead to suggestions that are too easy for User 1. On the other hand, recommending problems to User 3 based on User 1 could result in suggestions that are too difficult for User 3.

Similarly, if we consider User 2, who solved both Problem A and Problem B, a rating of 1 could suggest that Problem A and Problem B are similar, even though one may be more challenging than the other.

B. PROPOSED APPROACHES

In this paper three different approaches are proposed. The core concept of the proposed approaches, as illustrated in Figure 3, is to generate implicit user satisfaction ratings for programming problems, scaled from 1 to 5. These ratings take into account factors that the baseline approach does not consider, such as the user's current skill level and the problem's difficulty.

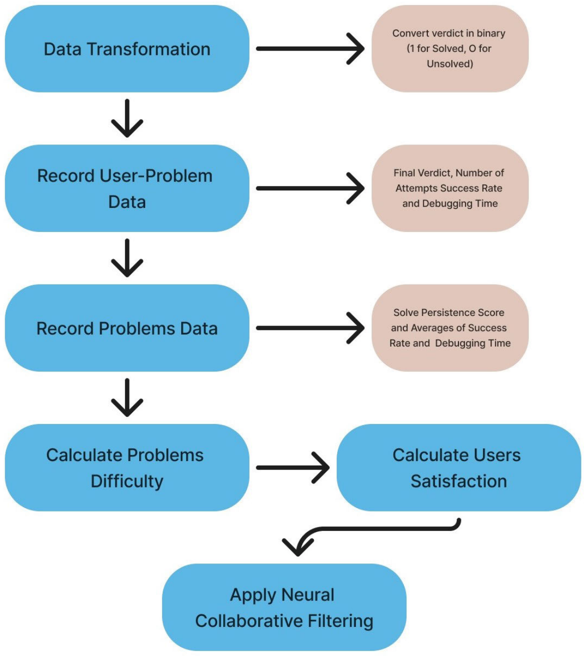
Data Transformation
Convert verdict in binary (1 for Solved, 0 for Unsolved)
Record User-Problem Data
Final Verdict, Number of Attempts Success Rate and Debugging Time
Record Problems Data
Solve Persistence Score and Averages of Success Rate and Debugging Time
Calculate Problems Difficulty
Calculate Users Satisfaction
Apply Neural Collaborative Filtering

FIGURE 3. Architecture of the proposed approaches.

The proposed approaches start by constructing a user-problem dataset that tracks each user's interactions with various programming problems. Additionally, problem-specific data is collected to characterize each problem. This problem data is used to assess the difficulty level of each problem by assigning a Difficulty score. User satisfaction is evaluated by analyzing the problem-solving patterns, considering both the difficulty of the problem and the user's current skill level. User satisfaction serves as an indicator of how users rate their experience when solving a problem, similar to how customer experiences are rated in business settings [51], allowing for a more accurate identification of similar users and problems, and ultimately improving the quality of recommendations and matches.

1) DATA TRANSFORMATION AND USER-PROBLEM DATA RECORD

Similar to the baseline approach, the proposed approaches also convert the final verdict (FV) into a binary value, with 1 indicating an 'Accepted' verdict and 0 for any other verdict. However, after this conversion, the proposed approaches additionally record the number of attempts each user required to solve each problem (NA). Furthermore, they capture the debugging time (DT), which is the elapsed time between the first and last submission. The success rate (SR) is also calculated, defined as the FV divided by NA. Table 3 shows how the proposed approaches transform the original submission log.

TABLE 3. Proposed approaches user-problem dataset.

$user$	$problem$	FV	NA	DT	SR
1	A	1	1	0	1
2	A	1	4	4000	0.25
3	A	1	1	0	1
2	B	1	1	0	1
4	B	0	1	0	0
5	B	0	3	3000	0

To ensure that all metrics are on a comparable scale, each metric is normalized. For a given metric $\mathbf{x}$, the normalized value is calculated as in equation 1.

$$Norm(x) = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{1}$$

In this equation, x is the original metric value, $\min(x)$ is the minimum value of the metric, and $\max(x)$ is the maximum value of the metric.

2) RECORD PROBLEMS DATA

The proposed approaches calculate the average values of DT (denoted as AVG_DT) and SR (denoted as AVG_SR) for each programming problem. The AVG_DT represents the average debugging time required by all users to solve a specific problem, providing insight into the typical amount of time users spend on that problem. The AVG_SR is the average success rate for each problem, calculated by averaging the success rates of all users who attempted the problem. Additionally, the Solve Persistence Score (SPS) is computed for each problem across all users. The SPS combines the age of a problem with the proportion of users who have successfully solved it, offering a balanced view of the problem's difficulty and persistence over time. The rationale behind the SPS is that older problems typically receive more exposure, and if they remain unsolved by a significant portion of the user base, it may suggest persistent difficulty. The SPS is calculated using the formula shown in equation 2. Table 4 shows the problem data derived from our example submission log.

$$SPS(p) = A(p) \times \frac{TU}{SU(p)} \tag{2}$$

In this equation, $A(p)$ represents the age of the problem, TU is the total number of users in the system, and $SU(p)$ is the number of users who have solved the problem p.

TABLE 4. Problem dataset.

$Problem$	AVG_DT	AVG_SR	SPS
A	1333.33	0.75	0.4740
B	1000	0.33	0.6907

C. DETERMINATION OF PROBLEM DIFFICULTY AND USER PERFORMANCE

The differences among the three approaches become evident in how they determine problem difficulty and assess user performance when solving a problem.

At the outset, it might appear straightforward to assess problem difficulty by observing the frequency of successful solutions, implying that problems with higher SRs are easier. However, a deeper analysis within OJs, which encompass both challenges and course exercises, reveals that this initial assumption is not consistently valid. Specifically, challenges, which are typically devised for competitive settings, are presumed to be more demanding than course exercises. Novices are more likely to engage with the simpler, course-focused problems and may face difficulties, necessitating multiple attempts to succeed, thereby lowering the SRs. Conversely, experienced programmers tend to tackle the more challenging problems directly and generally resolve them with greater efficiency, resulting in higher SRs for these more complex issues.

This insight indicates that a problem's SR alone does not reliably reflect difficulty level. Consequently, to achieve a more precise evaluation of problem difficulty on OJ systems, it is imperative to incorporate a wider array of variables beyond mere SRs.

In response to the complex nature of determining 'difficulty' in programming problems [52], our study aimed to develop a comprehensive way that integrates various factors influencing difficulty. Each approach uses the previously calculated variables (AVG_SR, AVG_DT, and SPS) to derive a meaningful difficulty score. Additionally, the problem type (T) is also considered. T is a binary variable that differentiates between course exercises and challenges, assigning a higher base difficulty to challenges to account for their inherently greater complexity.

The user performance (PU) aims to evaluate the process through which a user solves a problem, considering its difficulty level.

1) APPROACH 1: DIRECT WEIGHTED SUM

a: PROBLEM DIFFICULTY CALCULATION

This first approach is the most basic one, consisting of weighting each metric with the weights calculated previously. The difficulty score for each programming problem is calculated as in equation.3.

$$\mathrm{D} = W_1 \cdot AVG\_SR + W_2 \cdot AVG\_DT + W_3 \cdot SPS \tag{3}$$

W_1,W_2,W_3 are the weights of each metric.

b: USER PERFORMANCE CALCULATION

The formula provided in equation 4 calculates PU based on NA and DT, with DT_{min} representing the minimum DT in the User-Problem dataset.

$$PU = \frac{1}{NA} \times \frac{DT_{\min}}{DT} \tag{4}$$

In this formula, NA is used inversely ($\frac{1}{NA}$), meaning that fewer attempts result in a higher score, indicating better performance.

The ratio $\frac{DT_{min}}{DT}$ helps to scale the PU based on how quickly the user solves the problem compared to the minimum DT. If DT is close to $DT_{\min}$, the ratio approaches 1, indicating a performance close to the optimal DT.

2) APPROACH 2: FORMULA-BASED

a: PROBLEM DIFFICULTY CALCULATION

The difficulty D of a Programming Problem in this approach is calculated as equation 5.

$$\begin{aligned} D = {} & W_1 \cdot \log\left(1 + \frac{1}{AVG\_SR + \epsilon}\right) \\ & + W_2 \cdot AVG\_DT \\ & + W_3 \cdot \frac{1}{1 + \log\left(1 + Norm(SPS)\right)} \end{aligned} \tag{5}$$

In this equation, Norm(SPS) is the normalized value of SPS.

W_1,W_2,W_3 are the weights of each metric.

AVG_SR: The term $\log\left(1 + \frac{1}{AVG\_SR+\epsilon}\right)$ inversely correlates with difficulty. A lower SR indicates a more difficult problem, and as AVG_SR approaches 0, the difficulty score increases. The logarithmic function enhances differences at the lower end of the AVG_SR range, where problems are most challenging. To maintain numerical stability and prevent division by zero, a small constant (ϵ) is added to the AVG_SR calculations when it approaches zero.

AVG_DT: The term AVG_DT scales linearly with the debugging time, meaning problems requiring more time to solve contribute more to the difficulty score.

SPS: The term $\frac{1}{1+\log(1+\mathrm{Norm(SPS)})}$ inversely relates to difficulty, where a higher SPS suggests an easier problem. The logarithm smooths the effect of high SPS values, ensuring they do not disproportionately lower the difficulty score.

b: USER PERFORMANCE CALCULATION

The formula to calculate PU in this approach is given in equation 6.

$$P = \frac{1}{\log(NA + \alpha)} \times e^{\left(\frac{DT_{\min}}{DT+\beta}\right)} \tag{6}$$

The logarithmic function $\frac{1}{\log(NA+\alpha)}$ is used to reduce the impact of extreme values in NA, with α acting as a tuning parameter to balance penalties for multiple attempts. The exponential function $e^{\left(\frac{DT_{\min}}{DT+\beta}\right)}$ emphasizes performance improvement when the actual debugging time (DT) is close to the minimum time ($DT_{\min}$), while β prevents division by zero.

3) APPROACH 3: DISTANCE-BASED

In this approach, every problem or submission is considered a point in a multidimensional space defined its related metrics (AVG_SR, AVG_DT and SPS). Each metric represents a dimension in this space, and the values for a given problem or submission determine its coordinates in this multidimensional landscape.

a: PROBLEM DIFFICULTY CALCULATION

To determine each problem's difficulty, we introduced the notion of the Ideal Easy Problem (I). The I is characterized by optimal values for each metric, e.g., maximal AVG_SR and SPS, alongside minimal AVG_DT as shown in equation 7. This benchmark allows us to calculate the "distance" of each problem from the I using the Euclidean distance formula.

$$\begin{cases} AVG\_SR(I) = \max(AVG\_SR(P_i)) \\ SPS(I) = \max(SPS(P_i)) \\ AVG\_DT(I) = \min(AVG\_DT(P_i)) \end{cases} \tag{7}$$

In the above equation P_i represents each problem.

For each problem P_i and each metric M, the signed relative difference is calculated as in equation 8

$$Diff_M(P_i) = \frac{M(P_i) - M(I)}{M(I)} \tag{8}$$

The Euclidean distance $Dist(P_i)$ between each problem P_i and I is then calculated as in equation 9

$$Dist(P_i) = W_M \times \sqrt{\sum_M (Diff_M(P_i))^2} \tag{9}$$

In this equation, W_M represents the weight assigned to each metric.

b: USER PERFORMANCE CALCULATION

This approach treats each user submission and the average performance as points in a multidimensional space defined by the DT and NA. The Euclidean distance between these points provides a single measure of how different the user's performance is from the average.

For each performance metric, the signed relative difference is calculated to understand whether the user's performance is better or worse than the average as provided in equation 10.

$$Diff_M(U) = \frac{M(U) - M(AVG)}{M(U)} \tag{10}$$

In this equation, $M(U)$ represents the user's specific metric, while $M(AVG)$ denotes the average value of that metric among all users for the given problem.

Negative values indicate better performance than the average, while positive values indicate worse performance.

The Euclidean distance between these points, as calculated in equation 11, provides a single measure of how different the user's performance is from the average.

$$Dist(U) = \sqrt{Diff_{DT}(U)^2 + Diff_{NA}(U)^2} \tag{11}$$

In this equation, $Diff_{DT}(U)$ represents the difference in DT between the user and the average, while $Diff_{NA}(U)$ represents the difference in the NA between the user and the average.

To ensure that the positive or negative values impact the satisfaction, the normalized distance score is adjusted based on whether the signed relative differences are positive or negative. A negative signed relative difference (better performance) will reduce the normalized distance, while a positive signed relative difference (worse performance) will increase it.

For this adjustment, a simple factor that accounts for the proportion of negative differences is used.

$$Final\_Dist = \text{Norm}(Dist) \times \left(1 - \frac{ND}{TD}\right) \tag{12}$$

In this equation, ND represents the number of negative differences, and TD represents the total number of differences.

4) SCALING THE DIFFICULTY

To make difficulty scores interpretable and actionable, they are scaled to a familiar range of 1 to 5, where 1 represents 'Very Easy' and 5 represents 'Very Difficult'. The scaling process begins with normalizing the raw difficulty scores to a standard range of 0 to 1. After normalization, the scores are further refined to fit the 1 to 5 scale. A type-specific scaling approach is applied, where course-related problems (considered easier) are scaled using a quantile-based method, assigning scores between 1 and 3. Challenge-related problems (considered more difficult) are scaled similarly but within the range of 3 to 5. This ensures that difficulty scores accurately reflect problem types and their relative difficulty.

The generalized formula for scaling the difficulty scores is given in equation 13.

$$SD_i = Q(D_i) + 1 + 2T_i \tag{13}$$

In this equation, SD_i is the scaled difficulty score for problem i, $Q(D_i)$ is the quantile function that assigns the difficulty score D_i to one of three quantiles, and T_i indicates the problem type (0 for course-related, 1 for challenge-related).

D. USER SATISFACTION WITH PROGRAMMING PROBLEM

a: CONCEPTUALIZATION

In the realm of programming problems and OJs, the concept of user satisfaction is nuanced and distinct from traditional user ratings or reviews commonly found in other domains. Unlike products or services where users can readily provide explicit ratings or feedback, programming problems do not lend themselves to such straightforward satisfaction expressions. Instead, user satisfaction in this context is inferred and quantified to serve a specific purpose.

Programming problems are designed to challenge users' problem-solving skills and programming acumen. Users engage with these problems with the primary goal of finding solutions rather than explicitly rating their satisfaction with the problem itself. Consequently, there are no inherent mechanisms for users to provide subjective feedback or ratings as they would for a product or service.

The quantification of user satisfaction in the form of a rating serves as a crucial input for CF models.

b: COMPONENTS

The formula for calculating user satisfaction incorporates the Scaled SD, PU, the verdict, and the Skill Score (SU). These metrics were selected due to their direct relevance to the user's problem-solving efficiency and proficiency, while excluding others such for simplicity and clarity. This choice ensures that the formula remains interpretable and aligned with the user's experience in programming tasks.

The SR is not directly utilized in the user satisfaction calculation, as it does not distinguish between users who fail to solve a problem. Since the verdict is 0 for such users, the SR would always be 0 due to its calculation as the verdict divided by the number of attempts (NA). To address this limitation, the verdict and NA are used separately, offering a more nuanced understanding of the user's effort, persistence, and overall satisfaction.

c: SCALING THE SATISFACTION

To ensure meaningful and comparable metrics, the satisfaction score, like the difficulty score, is scaled from 1 to 5, where 1 represents the lowest satisfaction and 5 represents the highest. This scaling allows for consistent interpretation across different users and problem sets. Given the diverse skill levels in the system, ranging from novices to advanced programmers, it is crucial to account for these differences in calculating satisfaction. A novice, for example, may experience higher satisfaction from solving an intermediate-level problem compared to a more advanced programmer, who might find the same problem relatively easy and therefore less satisfying.

The formula for calculating satisfaction, provided in equation 14 takes these factors into account by incorporating the problem difficulty, the user's skill level, and performance metrics. This approach ensures that the satisfaction score reflects not only the outcome (i.e., solving the problem) but also the relative challenge posed to the user, based on their skill level. Thus, the satisfaction score is tailored to provide a more nuanced and fair assessment of user experience.

$$S(s) = \frac{SD}{2} \times v + 2.5 \times \min(PU \times \frac{SD}{SU}, 1) \tag{14}$$

In the provided formula, the satisfaction score $S(s)$ is calculated based on the scaled difficulty of the problem (SD), the verdict that the user got (v), the user performance (PU) and the user skill (SU)

$\frac{SD}{2} \times v$ represents the base satisfaction score derived from the problem's difficulty SD, scaled to a range from 0 to 2.5. The variable v indicates the verdict (with v=1 for solving the problem), ensuring that the base satisfaction reflects the challenge posed by the problem. A higher difficulty leads to a higher contribution to the satisfaction score.

$2.5 \times \min\left(PU \times \frac{SD}{SU}, 1\right)$ adjusts the satisfaction score based on the user's performance, encapsulated in PU. The ratio $\frac{SD}{SU}$ scales the difficulty SD relative to the user's skill level SU as calculated in equation 15, ensuring that more skilled users find easier problems less satisfying and vice versa. The min function caps this contribution to a maximum of 2.5, preventing the performance factor from disproportionately affecting the total satisfaction score.

$$SU = \frac{1}{k}\sum_{i=1}^{k} D_i \tag{15}$$

In the equation, k represents the number of the most challenging problems that a user has solved. Since users can progress from novice to advanced skill levels over time, including less challenging problems they initially solved could inaccurately reflect their current skill level. To address this, k is set to include only the top 10% most challenging problems the user has solved.

Having applied the designed satisfaction equation to the dataset, a 'User-Problem-Satisfaction Matrix' (Table 5) is created for each approach. This matrix effectively captures the satisfaction ratings for each user across different programming problems, based on problem difficulty and user performance. Its structured format, with users and problems systematically aligned, makes it well-suited for collaborative filtering (CF).

TABLE 5. Example of a user-problem-satisfaction matrix.

u	p	r
1	A	4
2	B	5
3	A	2
4	B	2

IV. EXPERIMENT

A. EXPERIMENTAL ENVIRONMENT

The experimental environment was set up on a desktop system featuring an Intel Core i7-12700K processor and 64 GB of RAM, running on Windows 10 Pro, version 22H2. An NVIDIA GeForce RTX 3070 Ti graphics card with 8 GB of dedicated GPU memory was utilized to accelerate machine learning tasks, particularly deep learning models. The software environment included Python, managed through the Anaconda distribution, which facilitated the installation and management of essential packages and dependencies. TensorFlow with GPU support was employed to optimize computational efficiency.

B. DATA

In this research, we delve into an extensive dataset derived from the Aizu Online Judge (AOJ), encompassing a total of 5870726 submissions made by 80,628 unique users across 2,752 different problems. AOJ serves as an interactive online platform that has been fostering programming skills since 2004. It's an educational hub where learners, from beginners to experts, engage with a diverse array of programming challenges [53], [54]. The platform, supporting an impressive range of 20 programming languages, offers over 3,000 distinct problems, thereby catering to a wide variety of learning needs and preferences [55].

C. COMPONENTS WEIGHTS IN DIFFICULTY FORMULA

Determining the appropriate weights for each component in difficulty calculation is essential for accurately assessing and comparing programming problems. Each metric captures a unique aspect of problem complexity but does not contribute equally to the overall difficulty. Assigning balanced weights ensures that the final difficulty score reflects the true nature of the problem and prevents any single metric from disproportionately affecting the assessment.

To derive these weights, we employ clustering and the Analytic Hierarchy Process (AHP). Clustering groups programming problems based on similarities across all metrics, allowing the analysis of cluster centroids to reveal typical metric values and varying difficulty levels. This data-driven method identifies the most significant metrics without needing predefined difficulty labels [56]. AHP complements clustering by incorporating expert judgment, using pairwise comparisons to determine the relative importance of each metric. Combining empirical data from clustering with expert-derived AHP weights provides a comprehensive and balanced approach to difficulty calculation.

We performed K-Means clustering on the programming problem dataset. K-Means is an unsupervised learning algorithm used to partition a dataset into K distinct, non-overlapping subsets (clusters) [57].

The algorithm begins with the initialization step, where K initial centroids are selected randomly from the dataset. Each data point is then assigned to the nearest centroid, forming K clusters. Mathematically, this assignment step can be expressed as shown in equation 16.

$$C_i = \arg\min_k \|x_i - \mu_k\|^2 \tag{16}$$

In the equation, C_i is the cluster assignment for data point x_i. It is the index of the cluster whose centroid is closest to x_i. The "arg min" operator finds the value of k (the cluster index) that minimizes the expression that follows. It returns the index k of the closest centroid to x_i. The term $\|x_i - \mu_k\|^2$ represents the squared Euclidean distance between the data point x_i and the centroid μ_k.

The Euclidean distance between two points x_i and mu_k in an m-dimensional space is defined as in equation 17.

$$\|x_i - \mu_k\|^2 = \sum_{j=1}^{m}(x_{ij} - \mu_{kj})^2 \tag{17}$$

In this equation x_{ij} is the j-th feature of the i-th data point, and μ_{kj} is the j-th feature of the k-th centroid.

After assigning all data points to clusters, the centroids are recalculated in the update step. The new centroid for each cluster is computed as the mean of all points assigned to that cluster, as shown in equation 18.

$$\mu_k = \frac{1}{|C_k|}\sum_{x_i \in C_k} x_i \tag{18}$$

In the above equation, $|C_k|$ is the number of points in cluster k, and x_i are the points assigned to cluster k. The assignment and update steps are repeated until the centroids no longer change significantly or a predefined number of iterations is reached.

The objective of K-Means is to minimize the within-cluster sum of squares (WCSS), also known as inertia [58]. The objective function is given by equation 19.

$$J = \sum_{k=1}^{K}\sum_{x_i \in C_k} \|x_i - \mu_k\|^2 \tag{19}$$

J represents the sum of squared distances between data points and their corresponding centroids.

To visualize the clustering results, we used Principal Component Analysis (PCA) to reduce the dimensionality of the data to two principal components. The resulting scatter plot, in Figure 4, shows three clusters, each represented by a different color. Clusters show significant overlap, indicating that the programming problems in these clusters share similar characteristics. However, since our primary goal is to find the weights for the metrics that contribute to problem difficulty, the lack of clear separation in the PCA space does not pose a significant issue. The analysis of cluster centers still provides valuable insights into the relative importance of each metric, which helps in deriving the necessary weights for an accurate difficulty calculation.

The centroids of clusters represent the average values of the metrics for all data points within each cluster [59]. Analyzing the centroids helps us understand the characteristics of each cluster, which in turn reveals insights into the difficulty levels of the programming problems.

To determine the weights for each metric, we first calculated the mean values of each metric across all clusters obtained from the K-Means algorithm. These mean values represent the central tendency of each metric within the clusters. Next, we normalize these mean values to ensure they sum to one, thereby providing a balanced representation of each metric's contribution to the overall difficulty score. The

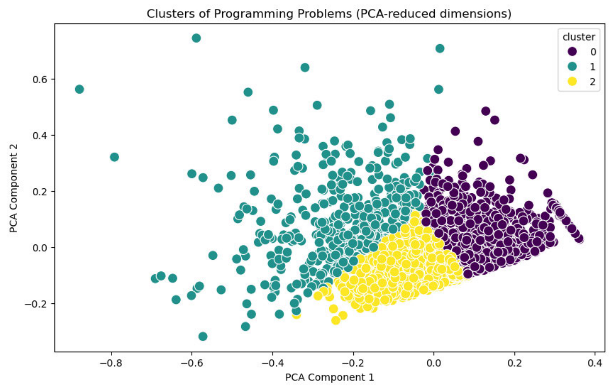

FIGURE 4. Programming problem clusters.

normalization process involves dividing each mean value by the sum of all mean values, as shown in equation 20.

$$w_j = \frac{\mu_j}{\sum_{j=1}^{m} \mu_j} \tag{20}$$

In the above equation, w_j is the normalized weight for metric j. μ_j is the mean value of metric j across all clusters and $\sum_{j=1}^{m} \mu_j$ is the sum of the mean values of all metrics.

The weights according to K-means are presented below in Table 6,

TABLE 6. Weights based on k-means.

Metric	Weight
AVG_DT	0.28
AVG_SR	0.61
SPS	0.11

AHP is a structured technique for organizing and analyzing complex decisions. AHP involves decomposing a decision problem into a hierarchy of more easily comprehensible sub-problems, each of which can be analyzed independently [60]. The AHP method combines both qualitative and quantitative aspects of decision-making and allows for a detailed assessment of the relative importance of various criteria [61].

The first step in AHP is to clearly define the problem and establish the goal of the decision-making process. In our case, the goal is to determine the weights for different metrics that contribute to the difficulty of programming problems.

Next, we structure the problem into a hierarchy. The top level of the hierarchy is the overall goal. The subsequent levels break down the criteria and sub-criteria that will influence the decision. In our case, the criteria are the metrics i.e. AVG_SR, AVG_DT, and SPS.

For each level of the hierarchy, we construct pairwise comparison matrices. These matrices allow us to compare the relative importance of each criterion against the others. The comparisons are made using a scale of relative importance that ranges from 1 (equal importance) to 9 (extreme importance of one element over another). Based on expert consultation, the pairwise comparison matrix for our metrics is presented in Table 7

TABLE 7. Pairwise comparison matrix for AHP.

	AVG_SR	AVG_DT	SPS
AVG_SR	1	2	1
AVG_DT	1/2	1	1
SPS	1	1	1

To ensure the comparisons are consistent, we calculate the consistency ratio (CR). The CR is a measure of how consistent the judgments have been relative to large samples of purely random judgments. A CR less than 0.1 is generally considered acceptable [62]. To calculate the CR, we first calculated the priority vector (PV) which consist on dividing each element by the sum of its column, and then average the rows as provided in equation 21. Then, we calculated the consistency index (CI) by comparing the maximum eigenvalue ($\lambda_{\max}$), as shown in equation 22, of the matrix to the number of items being compared. The eigenvalue is a special number that shows how much a certain transformation (represented by a matrix) stretches or shrinks a vector. The CI is calculated as in equation 23.

$$\text{Priority Vector} = \left(\frac{1}{n} \sum_{j=1}^{n} \frac{a_{1j}}{\sum_{k=1}^{n} a_{kj}}, \ldots, \frac{1}{n} \sum_{j=1}^{n} \frac{a_{nj}}{\sum_{k=1}^{n} a_{kj}} \right) \tag{21}$$

In this equation a_{ij} represents the element in the i-th row and j-th column of the normalized pairwise comparison matrix, and n is the number of items being compared.

$$\lambda_{\max} = \frac{1}{n} \sum_{i=1}^{n} \frac{(\mathrm{PV}_i \cdot A \cdot \mathrm{PV})_i}{\mathrm{PV}_i} \tag{22}$$

In this equation, PV is the priority vector and A is the pairwise comparison matrix.

$$CI = \frac{\lambda_{\max} - n}{n - 1} \tag{23}$$

The CR, as shown in equation 24, is calculated by dividing the CI by the random index (RI), which depends on the number of items being compared. The RI values are pre-determined for matrices of different sizes. For n=3, RI = 0.5247.

$$CR = \frac{CI}{RI} \tag{24}$$

With a CR value of 0.056, which is less than 0.1, the matrix is considered to be consistent.

The corresponding weights are provided in table 8.

The final weights, which are the average of K-means approach and AHP, are show in table 9.

TABLE 8. **Weights based on AHP.**

Metric	Weight
AVG_SR	0.413
AVG_DT	0.26
SPS	0.327

TABLE 9. **Final weights for each metric.**

Metric	Final Weight
AVG_SR	0.51
AVG_DT	0.27
SPS	0.22

D. NEURAL COLLABORATIVE FILTERING IMPLEMENTATION

The process completed so far has enabled us to create three user-problem-satisfaction matrices, each based on a different approach [63]. The final goal is to apply NCF model to each of these matrices. NCF is a modern approach in RSs that leverages the power of neural networks to model complex and non-linear user-item interactions. Unlike traditional methods that rely on linear assumptions, NCF can capture more intricate patterns in the data, leading to more accurate and personalized recommendations [64].

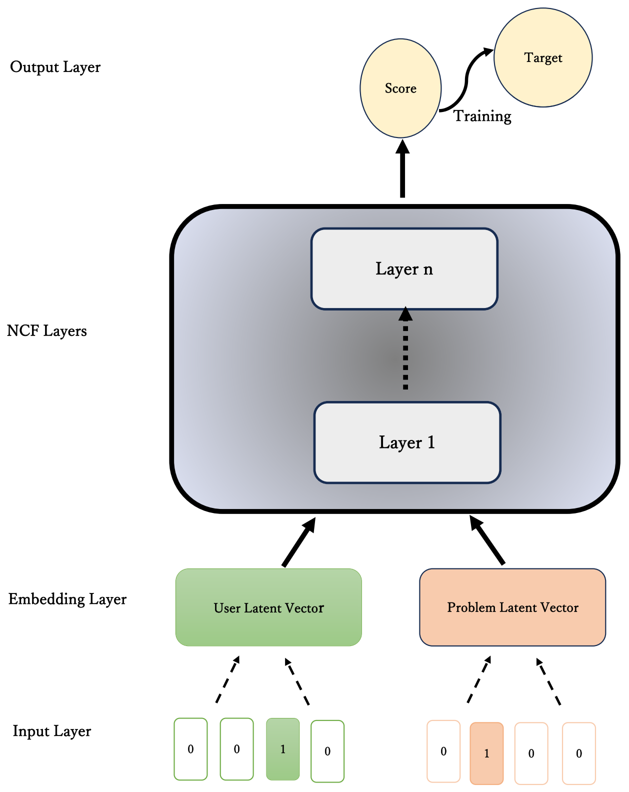

FIGURE 5. **Structure of the NCF.**

Figure 5 shows the structure of the NCF as used in our methodology. At the foundational level, the input layer is composed of two distinct arrays representing users and problems within the system. These arrays are encoded using one-hot encoding technique, which converts categorical data into a numerical format that can be understood by machine learning algorithms.

Progressing upwards, the embedding layer acts as a critical transformational stage. As show in equations 25 and 26, the sparse one-hot vectors undergo a form of compression, transitioning into dense, lower-dimensional vectors known as latent vectors [65]. These vectors are rich with latent features, representing unobserved user preferences and problems attributes distilled from the input data. In this capacity, the embedding layer serves to distill and concentrate the raw input data into a more potent and informative format, setting the stage for the subsequent layers to uncover patterns and relationships.

$$\mathbf{x_u} = f(\mathbf{W_u} \cdot \mathbf{u} + \mathbf{b_u}) \tag{25}$$

$$\mathbf{x_p} = f(\mathbf{W_p} \cdot \mathbf{p} + \mathbf{b_p}) \tag{26}$$

where $\mathbf{W_u}$ and $\mathbf{W_p}$ are the weight matrices, $\mathbf{b_u}$ and $\mathbf{b_p}$ are the bias vectors, and f is a non-linear activation function. The activation function employed is the Rectified Linear Unit (ReLU), as defined in equation 27.

$$f(x) = \max(0, x) \tag{27}$$

In equation 27, if the input x is positive, the function will output x, and if x is negative, the function will output zero. The ReLU function is favored in many neural network applications because it introduces non-linearity into the model while mitigating the vanishing gradient problem.

The central components of this architecture are the NCF layers, which consist of multiple levels of neural networks, each designed to capture varying degrees of interaction complexity [66]. These layers function by taking the user and problems latent vectors $\mathbf{x_u}$ and $\mathbf{x_p}$ and processing them through a series of transformations as shown in equation 28. As data passes through these layers, the model learns to discern and represent the complex and non-linear relationships that exist between users and problems.

$$\mathbf{h} = f(\mathbf{W_h} \cdot [\mathbf{x_u}; \mathbf{x_p}] + \mathbf{b_h}) \tag{28}$$

where $[\mathbf{x_u}; \mathbf{x_p}]$ denotes the concatenation of the user and problem latent vectors, $\mathbf{W_h}$ is the weight matrix for the hidden layers, $\mathbf{b_h}$ is the bias vector for the hidden layers, and h represents the output of the hidden layers.

Culminating at the peak of the structure is the output layer. This layer synthesizes the insights gleaned from the NCF layers and articulates them as a singular score $\hat{y}$ as calculated in the equation 29. This score represents the predicted likelihood of a user's preference for a problem, which is the cornerstone of the model's recommendation capability [67]. The output is typically a real-valued number that can be thresholded or ranked to suggest the most appropriate problems to a user.

$$\hat{y} = \sigma(\mathbf{w_o}^T \cdot \mathbf{h} + b_o) \tag{29}$$

where σ is the sigmoid activation function, $\mathbf{w_o}$ is the weight vector for the output layer, b_o is the bias for the output layer, and $\hat{y}$ is the predicted score.

An essential part of this model's utility is its training mechanism, where the model learns from known user-problems interactions. The model is trained by adjusting its parameters to minimize the loss L between the predicted scores and the actual user interactions, often referred to as the target scores. This training process is iterative, involving forward propagation of inputs to generate scores, comparison against actual targets, and backward propagation of errors to adjust the model's parameters. The loss function used is mean squared error calculated as in shown in the equation 30

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{30}$$

where N is the number of user-problem interaction pairs in the training set.

The model is a hybrid collaborative filtering (CF) approach that combines the strengths of both user-based and item-based methods to improve recommendation accuracy and relevance [63]. Separate latent spaces for users and items are created through embedding layers, capturing complex factors that influence user preferences and item characteristics.

After generating these embeddings, the model combines the user and item vectors into a single feature set, representing their interaction. This allows the model to consider both the similarities between users or items, as in traditional CF methods, and the unique relationships between specific users and items. The concatenated vectors are passed through dense layers, adding non-linearity and enabling the model to better understand user-item interactions. The final output is a prediction that reflects both user behavior and item affinity.

This hybrid method overcomes the limitations of traditional CF techniques, addressing issues where user-based methods struggle to recommend new items and where item-based methods miss individual user preferences.

V. RESULTS

A. PROBLEM DIFFICULTY

The Pearson correlation coefficient, as calculated in equation 31, measures the linear relationship between two variables, producing a value between −1 and 1. A value of 1 implies a perfect positive linear relationship, −1 implies a perfect negative linear relationship, and 0 implies no linear relationship [68]. It is calculated using the covariance of the variables divided by the product of their standard deviations. The formula is:

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}} \tag{31}$$

In the above equation, x_i and y_i are the individual sample points, and $\bar{x}$ and $\bar{y}$ are the mean values of x and y, respectively.

The Pearson correlation coefficients presented in Figure 6 highlight the strong alignment between Approach 1 and Approach 3 (0.85), showcasing their close similarity in assessing problem difficulty. The correlation between Approach 1 and Approach 2 is also promising (0.71), reflecting a significant level of consistency while acknowledging their unique perspectives. Similarly, the correlation between Approach 2 and Approach 3 (0.73) underscores a meaningful relationship, demonstrating a balanced blend of shared insights and complementary sensitivities.

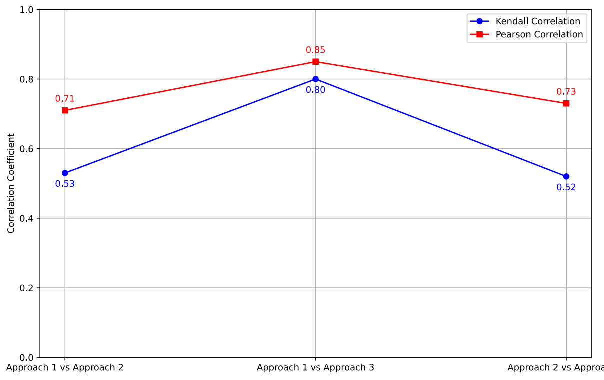

FIGURE 6. Correlation Coefficients For Problems Difficulty.

Further analysis using Kendall correlation coefficients, as calculated in equation 32 and depicted in Figure 6, reveals a strong ordinal association between Approach 1 and Approach 3, with a coefficient of 0.80. The correlation between Approach 1 and Approach 2, as well as between Approach 2 and Approach 3, are moderate, with coefficients of 0.53 and 0.52, respectively. This consistency across different correlation measures underscores that while all three approaches share common elements, Approach 1 and Approach 3 are more closely aligned in ranking the problems, suggesting a stronger agreement in their difficulty assessment.

$$\tau = \frac{C - D}{\sqrt{(C + D + T) \cdot (C + D + U)}} \tag{32}$$

in this equation, C is the number of concordant pairs, D is the number of discordant pairs, T is the number of ties in the first variable, and U is the number of ties in the second variable.

B. USER SATISFACTION

The Pearson correlation coefficients presented in Figure 7 show a strong positive linear relationship between Approach 1 and Approach 3 (0.78), indicating that these methods are closely aligned in their assessment of user satisfaction. The correlation between Approach 1 and Approach 2 is moderate (0.66), suggesting some consistency but also highlighting differences. The correlation between Approach 2 and Approach 3 (0.75) also reflects a strong relationship, implying these approaches are relatively consistent in their sensitivity.

Further analysis using Kendall correlation coefficients, depicted in Figure 7, highlights a meaningful ordinal

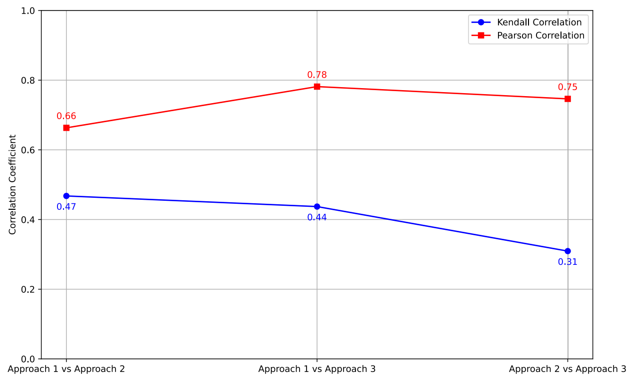

FIGURE 7. Correlation Coefficients For User Satisfaction.

association between Approach 1 and Approach 3, with a coefficient of 0.44. The correlation between Approach 1 and Approach 2 is similarly moderate (0.47), reflecting a solid level of agreement in their rankings. The correlation between Approach 2 and Approach 3, though lower at 0.31, suggests that each approach offers unique perspectives, contributing to a diverse understanding of the problem's characteristics.

C. MODEL PERFORMANCE

Alongside the three proposed approaches, we included a baseline dataset that categorizes exercises in a straightforward binary format, marking them as either completed (1) or not completed (0). This baseline model acts as a basic reference point, allowing us to evaluate the effectiveness of the three more complex methods by comparing them to this simple, widely used approach.

We divided each dataset into three parts to effectively train and evaluate our model. Initially, we separated 80% of the data to form the training set, which is used to teach the model. The remaining 20% is held back as the test set, reserved for the final evaluation of the model. To fine-tune the model and check its performance during the training phase, we further set aside 20% of the training set as a validation set. This approach ensures that our model learns from the majority of the data, while we maintain separate datasets to adjust model settings and to test its ultimate accuracy on data it has never seen before.

Table 10 and Figure 8 present the root squared mean error (RMSE) across the four different approaches. The baseline approach shows the lowest RMSE at 0.0036, indicating minimal error in predictions, while Approach 1 has an RMSE of 0.03, and Approaches 2 and 3 have RMSEs of 0.031 and 0.033, respectively. This suggests that the baseline approach fits the training data best, though this might also indicate overfitting due to its simplistic binary nature.

The RMSE values for testing are slightly higher across all methods, with the baseline at 0.009 and Approaches 1, 2, and 3 at 0.031, 0.033, and 0.031, respectively. The increase in RMSE from training to testing indicates that the models may be slightly overfitting the training data but still maintain reasonable performance when exposed to new data.

Validation RMSEs are the highest among the three metrics, with the baseline at 0.012 and Approaches 1, 2, and 3 at 0.035, 0.042, and 0.035, respectively. This further confirms the generalization capability of each model. The lower RMSE in the baseline approach might again be due to the simplistic nature of the binary classification, while the other approaches show slightly higher errors due to the more complex nature of their predictive tasks.

TABLE 10. RMSE Values Across Different Approaches.

Phase	Baseline	Approach 1	Approach 2	Approach 3
Training	0.0036	0.030	0.031	0.033
Testing	0.009	0.031	0.033	0.031
Validation	0.012	0.035	0.042	0.035

To enhance our understanding of each approach, we examine 5 key performance metrics: Precision (P), Recall(R), Hit Rate (HR), Mean Reciprocal Rank (MRR) and the Coverage (C). These metrics, as presented in Table 11, offer insights into the system's prediction accuracy and its comprehensive ability to identify all pertinent items.

TABLE 11. Performance metrics of different approaches.

Metric	Baseline	Approach 1	Approach 2	Approach 3
Precision	0.88	0.93	0.86	0.93
Recall	0.89	0.94	0.87	0.94
Hit Score	1	1	1	1
MRR	0.12	0.40	0.52	0.49
Coverage	0.23	0.37	0.40	0.41

Precision, as calculated in equation 33 assesses the accuracy of the system's positive predictions. It is defined as the proportion of items correctly identified as relevant (true positives) out of all items the system has classified as relevant [69]. This metric is crucial for understanding how many of the recommended items are actually of interest to the user.

$$P = \frac{TP}{TP + FP} \tag{33}$$

In this equation, TP represents the number of true positives, and FP denotes the number of false positives.

Figure 9 shows the precision values on the test set. All approaches demonstrated precision values above a satisfactory level, suggesting they effectively enhance the relevance of recommended items compared to the baseline. Approach 1 and Approach 3 achieved the highest precision at 0.93, indicating they are the most effective in delivering relevant recommendations. The Baseline method, with a precision of 0.88, and Approach 2, with a precision of 0.86, still indicate good levels of recommendation relevance, though to a lesser extent.

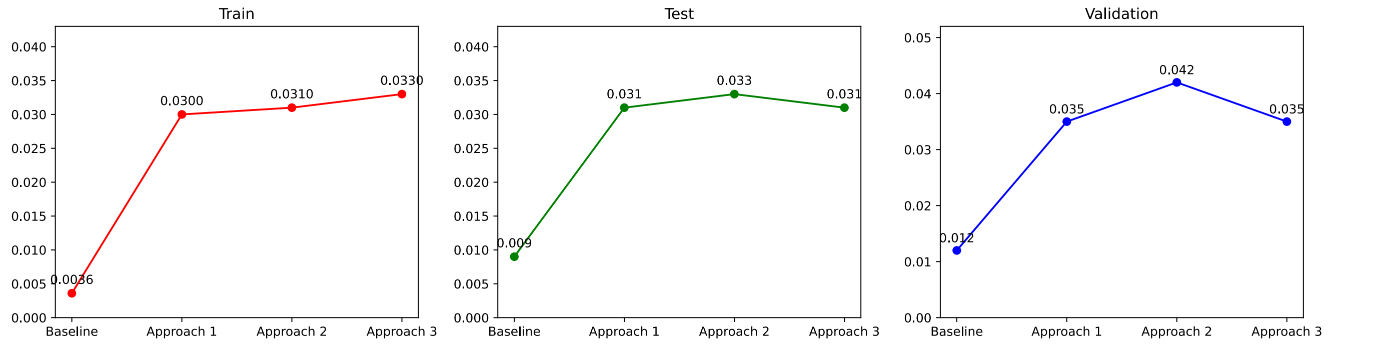

FIGURE 8. Loss Function across different datasets and approaches.

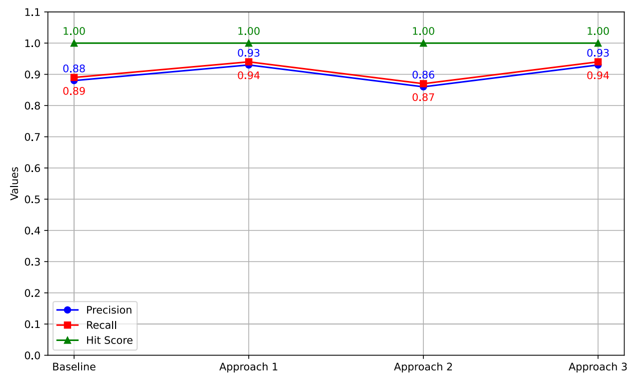

FIGURE 9. Precision, Recall and Hit Score.

Recall, on the other hand, as calculated in equation 34 measures the system's capability to identify all relevant items. It quantifies the proportion of actual relevant items that have been correctly recommended by the system [70]. Recall is vital for ensuring that the system does not overlook significant items.

$$R = \frac{TP}{TP + FN} \tag{34}$$

In this equation, FN represents the number of true negatives.

As shown in Figure 9, all approaches demonstrate strong recall values, indicating a high ability to identify relevant items. Approaches 1 and 3 both achieved the highest recall at 0.94, demonstrating their equal effectiveness in retrieving relevant items. The Baseline approach also performed well, with a recall of 0.89, though slightly lower than that of Approaches 1 and 3. Approach 2, while still achieving a respectable recall of 0.87, exhibited the lowest recall among the methods compared.

Hit Rate(HR), as calculated in equation 35 measures the proportion of users for whom at least one relevant item is included in the recommended list [71].

$$HR = \frac{NH}{TU} \tag{35}$$

In this equation, NH represents the number of users with at least one hit and TU the total number of users.

All the approaches in the study demonstrated a hit score of 1, indicating that for each query or user interaction, the top-ranked item in the recommendation list was relevant. A hit score of 1 signifies that the model consistently includes at least one relevant item in the top position of its recommendations across all test cases. This metric reflects the model's effectiveness in ensuring that users are presented with valuable content immediately, which is crucial for user satisfaction and engagement. The consistent hit score across all approaches suggests that they are all capable of accurately identifying and prioritizing relevant items in their predictions.

The Mean Reciprocal Rank (MRR) as calculated is equation 36 is used to evaluate the quality of ordered lists, often in the context of information retrieval or RSs. It measures the rank of the first relevant item in the list for each query and computes the average of the reciprocal ranks [72].

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i} \tag{36}$$

in the above equation, $|Q|$ is the total number of users and $rank_i$ is the rank position of the first relevant item in the list of recommendations for the i-th user.

The results show that all the new approaches significantly enhance the ranking effectiveness compared to the Baseline model. The low MRR value of 0.12 for the Baseline model indicates poor performance in ranking relevant items highly. In contrast, Approach 1, with an MRR of 0.40, demonstrates a notable improvement, indicating that relevant items are ranked much higher on average. Approach 2, with the highest MRR value of 0.52, proves to be the most effective in ensuring users are quickly presented with the most relevant items, significantly improving user satisfaction and engagement. Approach 3 also performs well, with an MRR of 0.49, better than both the Baseline and Approach 1, but slightly less effective than Approach 2.

The Coverage (Cv) is a metric that measures the proportion of items in the dataset that have been recommended to users.

It provides an indication of how well a system is exploring the catalog of items [73]. High coverage suggests that the system is recommending a wide variety of items, whereas low coverage indicates that recommendations are concentrated on a smaller subset of items.

$$Cv = \frac{UI}{TI} \tag{37}$$

In this equation, UI represents the number of unique items recommended and TI the total number of items in the catalog.

The coverage values reveal that all new approaches (Approach 1, Approach 2, and Approach 3) significantly enhance the diversity of recommendations compared to the Baseline. The Baseline model, with a coverage of 0.23, focuses on a limited subset of items, which may restrict user engagement and satisfaction. In contrast, Approach 1 (0.37), Approach 2 (0.40), and Approach 3 (0.41) show considerable improvements, indicating their ability to recommend a wider variety of items from the catalog. Among the approaches, Approach 2 stands out with the highest coverage, suggesting it is the most effective at promoting item diversity. Approach 3 also performs well, closely following Approach 2, while Approach 1 shows a significant but slightly lower improvement.

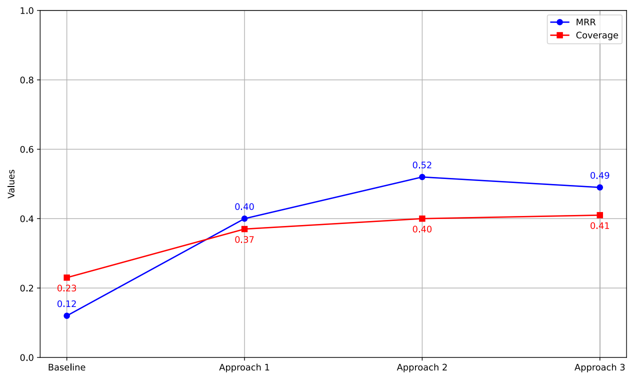

FIGURE 10. MRR and coverage.

VI. DISCUSSION

A. ERROR FUNCTION PERFORMANCE

The baseline approach performed better on the error function, primarily because it uses values between 0 and 1. This narrow range allows for more precise error calculation, resulting in lower overall error values. In contrast, the proposed approaches operate with values ranging between 1 and 5, which naturally leads to larger error magnitudes. However, it is important to note that despite this broader range, the error values for the proposed approaches remain within highly acceptable limits. This suggests that while the baseline approach has a slight edge in terms of error function performance, the proposed approaches still maintain robust performance levels.

B. ACCURACY

When evaluating accuracy, which includes precision, recall, and hit score, the baseline approach demonstrates strong performance, particularly in comparison to Approach 2. Precision measures the proportion of true positive recommendations out of all positive recommendations made, while recall measures the proportion of true positive recommendations out of all actual positive cases. The hit score measures the rate at which relevant items are found among the top recommendations. The baseline approach excels in these metrics, indicating that it is effective in accurately recommending relevant items. However, Approaches 1 and 3 surpass the baseline in both precision and recall, suggesting that these methods are better at identifying and recommending relevant items.

C. QUALITY OF RECOMMENDED ITEMS

A significant difference between the approaches is evident in the quality of the recommended items. The *MRR* values show that the proposed approaches significantly outperform the baseline approach. Higher *MRR* values indicate that relevant items are presented to users earlier in the recommendation list. This early presentation is crucial for user satisfaction, as it ensures that users can quickly find items of interest. The proposed approaches, with their higher *MRR* values, demonstrate a clear advantage in delivering high-quality recommendations, thereby enhancing the overall user experience.

Coverage is another important metric that differentiates the baseline approach from the proposed methods. The baseline approach recommends only a small subset of problems to users, with a coverage value of 0.23. This limited coverage means that users are exposed to a narrow range of items, which can lead to repetitive recommendations and reduced user engagement. In contrast, the proposed approaches show significantly higher coverage values, with Approach 2 achieving the highest coverage at 0.43. This means that the proposed methods recommend a much broader range of items from the catalog, increasing diversity and potentially boosting user satisfaction by offering a wider variety of choices.

D. ADAPTABILITY TO USER SKILL AND PROBLEM DIFFICULTY

One of the notable advantages of the proposed approaches is their ability to incorporate problem difficulty and user skill level into the recommendation process. This adaptability allows the system to tailor recommendations to the user's current skill level, ensuring that the recommended problems are appropriately challenging. The baseline approach, on the other hand, only considers whether a user has solved a problem or not. This can lead to recommendations that do not match the user's evolving skill level, such as suggesting problems that are too easy for an experienced user. The proposed approaches provide more personalized and relevant recommendations, which can better support user learning and progression.

E. PERFORMANCE ESTIMATION FOR RECOMMENDED PROBLEMS

Another significant advantage of the proposed approaches is their ability to estimate the user's performance on recommended problems. This feature allows the system to inform users about the expected time or number of attempts needed to solve a problem. Such information can help users manage their time and set realistic expectations, enhancing the overall user experience. The baseline approach lacks this capability, as it focuses solely on whether similar problems have been solved by other users. The performance estimation provided by the proposed approaches adds an additional layer of user support, making the RS more dynamic and helpful.

F. BROADER EDUCATIONAL APPLICATIONS

The proposed approaches can be applied to a wide range of educational disciplines beyond programming to improve learning outcomes and student engagement. These methods can help educators assess the difficulty of tasks across different subjects, estimate students' skill levels, and evaluate their performance based on measurable factors such as time spent on a task and the number of attempts required to complete it.

These approaches can offer a deeper understanding of both the content being taught and the individual needs of learners. For example, in mathematics, the system can calculate the difficulty of a problem and can evaluate students' problem-solving abilities in real-time. In language learning, it can assess reading or comprehension skills and provide exercises that match the student's proficiency level. This flexibility can make these methods valuable in various educational settings, ensuring that students can receive the appropriate level of challenge and support.

VII. CONCLUSION AND FUTURE WORK

In this study, we proposed different approaches to improve the recommendation of programming problems on an Online Judge (OJ) platform. Unlike traditional CF methods that rely on explicit user ratings, our approaches transform implicit interaction data into a structured rating system, simulating explicit feedback. This innovative methodology allows the RS to better cater to the diverse needs of learners.

We implemented NCF with the proposed approaches and also used a baseline approach based on a binary system. The baseline approach performed better on the error function due to its use of values between 0 and 1. However, the proposed approaches, which use values from 1 to 5, maintained acceptable error values, showing strong performance.

In terms of accuracy, including precision, recall, and hit score, the baseline approach was effective, especially compared to Approach 2. However, Approaches 1 and 3 performed better, demonstrating their ability to recommend relevant items accurately.

The quality of recommendations significantly improved with the proposed approaches. The Mean Reciprocal Rank (MRR) values showed that the proposed methods outperformed the baseline, presenting relevant items to users earlier. The proposed approaches also excelled in coverage, recommending a broader range of items compared to the baseline. This diversity enhances user satisfaction by offering more varied choices.

A key advantage of the proposed approaches is their ability to consider problem difficulty and user skill level, providing personalized recommendations. Additionally, they can estimate the user's performance on recommended problems, helping users manage their time and set realistic expectations.

Our contributions mark an advancement in RSs, particularly in educational technology. The approaches offer precise, personalized, and diverse problem recommendations, demonstrating the effectiveness of combining various metrics in RS design. Future efforts will focus on refining the model's sophistication, incorporating richer data sources, exploring source code instructions, and integrating advanced algorithms to further enhance the learning experience. The ultimate goal is to deploy these approaches on a live OJ platform, evaluate their performance in a real educational context, and use the insights gained to improve the system even further.

REFERENCES

[1] C. G. Demartini, L. Benussi, V. Gatteschi, and F. Renga, "Education and digital transformation: The 'riconnessioni' project," *IEEE Access*, vol. 8, pp. 186233–186256, 2020.

[2] S. Mhlongo, K. Mbatha, B. Ramatsetse, and R. Dlamini, "Challenges, opportunities, and prospects of adopting and using smart digital technologies in learning environments: An iterative review," *Heliyon*, vol. 9, no. 6, Jun. 2023, Art. no. e16348.

[3] J. Govea, E. O. Edye, S. Revelo-Tapia, and W. Villegas-Ch, "Optimization and scalability of educational platforms: Integration of artificial intelligence and cloud computing," *Computers*, vol. 12, no. 11, p. 223, Nov. 2023.

[4] M. M. Rahman, Y. Watanobe, T. Matsumoto, R. U. Kiran, and K. Nakamura, "Educational data mining to support programming learning using problem-solving data," *IEEE Access*, vol. 10, pp. 26186–26202, 2022.

[5] D. M. Muepu, Y. Watanobe, and M. M. Rahman, "Collaborative filtering based on non-negative matrix factorization for programming problem recommendation," in *Advances and Trends in Artificial Intelligence. Theory and Applications*. Cham, Switzerland: Springer, 2023, pp. 241–250.

[6] M. M. Rahman and Y. Watanobe, "Multilingual program code classification using *n*-layered bi-LSTM model with optimized hyperparameters," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 8, no. 2, pp. 1–17, Apr. 2024.

[7] C. M. Intisar, Y. Watanobe, M. Poudel, and S. Bhalla, "Classification of programming problems based on topic modeling," in *Proc. 7th Int. Conf. Inf. Educ. Technol.*, New York, NY, USA, Mar. 2019, pp. 275–283.

[8] D. Bilegjargal and N.-L. Hsueh, "Understanding students' acceptance of online judge system in programming courses: A structural equation modeling approach," *IEEE Access*, vol. 9, pp. 152606–152615, 2021.

[9] F. Iffath, A. S. M. Kayes, M. T. Rahman, J. Ferdows, M. S. Arefin, and M. S. Hossain, "Online judging platform utilizing dynamic plagiarism detection facilities," *Computers*, vol. 10, no. 4, p. 47, Apr. 2021.

[10] N.-L. Hsueh, L.-C. Lai, and W.-H. Tseng, "Design of an online programming platform and a study on Learners' testing ability," *Electronics*, vol. 12, no. 22, p. 4596, Nov. 2023.

[11] M. Rahman, Y. Watanobe, P. Szmeja, P. Sowinski, M. Paprzycki, and M. Ganzha, "Code semantics learning with deep neural networks: An AI-based approach for programming education," in *Computational Science—ICCS 2023* (Lecture Notes in Computer Science). Cham, Switzerland: Springer, 2023, pp. 737–750.

[12] T. Saito and Y. Watanobe, "Learning path recommendation system for programming education based on neural networks," *Int. J. Distance Educ. Technol.*, vol. 18, no. 1, pp. 36–64, Jan. 2020.
[13] P. Fantozzi and L. Laura, "A dynamic recommender system for online judges based on autoencoder neural networks," in *Proc. Int. Conf. Methodologies Intell. Syst. Techhnol. Enhanced Learn.*, Cham, Switzerland: Springer, 2021, pp. 197–205.
[14] B. Xu, S. Yan, X. Jiang, and S. Feng, "SCFH: A student analysis model to identify Students' programming levels in online judge systems," *Symmetry*, vol. 12, no. 4, p. 601, Apr. 2020.
[15] F. Zaffalon, A. Prisco, R. De Souza, D. Teixeira, W. Paes, P. Evald, N. Tonin, S. Devincenzi, and S. Botelho, "A recommender system of computer programming exercises based on student's multiple abilities and skills model," in *Proc. IEEE Frontiers Educ. Conf. (FIE)*, Oct. 2022, pp. 1–8.
[16] M. Gotthardt and V. Mezhuyev, "Measuring the success of recommender systems: A PLS-SEM approach," *IEEE Access*, vol. 10, pp. 30610–30623, 2022.
[17] N. Torres, "Recommender systems for education: A case of study using formative assessments," in *Proc. 41st Int. Conf. Chilean Comput. Sci. Soc. (SCCC)*, Nov. 2022, pp. 1–6.
[18] J. Ben Schafer, D. Frankowski, J. Herlocker, and S. Sen, *Collaborative Filtering Recommender Systems*. Berlin, Germany: Springer, 2007, pp. 291–324.
[19] S. Sopchoke and B. Kijsirikul, "A step towards high quality one-class collaborative filtering using online social relationships," in *Proc. Int. Conf. Adv. Comput. Sci. Inf. Syst.*, Dec. 2011, pp. 243–248.
[20] R. Chen, K. Pang, M. Huang, H. Liang, S. Zhang, L. Zhang, P. Li, Z. Xia, J. Zhang, and X. Kong, "A survey on recommendation methods based on social relationships," *Electronics*, vol. 12, no. 22, p. 4564, Nov. 2023.
[21] A. Fareed, S. Hassan, S. B. Belhaouari, and Z. Halim, "A collaborative filtering recommendation framework utilizing social networks," *Mach. Learn. With Appl.*, vol. 14, Dec. 2023, Art. no. 100495.
[22] I. Gligorea, M. Cioca, R. Oancea, A.-T. Gorski, H. Gorski, and P. Tudorache, "Adaptive learning using artificial intelligence in e-learning: A literature review," *Educ. Sci.*, vol. 13, no. 12, p. 1216, Dec. 2023.
[23] N. Torres, "A multimodal user-adaptive recommender system," *Electronics*, vol. 12, no. 17, p. 3709, Sep. 2023.
[24] K. Yan, "A review of techniques used in e-commerce recommendation system," *Appl. Comput. Eng.*, vol. 4, no. 1, pp. 629–635, May 2023.
[25] R. Sharma and R. Singh, "Evolution of recommender systems from ancient times to modern era: A survey," *Indian J. Sci. Technol.*, vol. 9, no. 20, pp. 1–12, May 2016.
[26] N. Yanes, A. M. Mostafa, M. Ezz, and S. N. Almuayqil, "A machine learning-based recommender system for improving students learning experiences," *IEEE Access*, vol. 8, pp. 201218–201235, 2020.
[27] B. Xiao and I. Benbasat, "E-commerce product recommendation agents: Use, characteristics, and impact," *MIS Quart.*, vol. 31, no. 1, p. 137, 2007.
[28] Z. Fan, D. Chang, and J. Cui, "Algorithm in e-commerce recommendation," in *Proc. 5th Int. Conf. Ind. Econ. Syst. Ind. Secur. Eng. (IEIS)*, Aug. 2018, pp. 1–6.
[29] D. Roy and M. Dutta, "A systematic review and research perspective on recommender systems," *J. Big Data*, vol. 9, no. 1, p. 59, May 2022.
[30] A. Gatzioura, J. Vinagre, A. M. Jorge, and M. Sànchez-Marrè, "A hybrid recommender system for improving automatic playlist continuation," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 5, pp. 1819–1830, May 2021.
[31] F. Wang, H. Zhu, G. Srivastava, S. Li, M. R. Khosravi, and L. Qi, "Robust collaborative filtering recommendation with user-item-trust records," *IEEE Trans. Computat. Social Syst.*, vol. 9, no. 4, pp. 986–996, Aug. 2022.
[32] Z. Cui, X. Xu, F. Xue, X. Cai, Y. Cao, W. Zhang, and J. Chen, "Personalized recommendation system based on collaborative filtering for IoT scenarios," *IEEE Trans. Services Comput.*, vol. 13, no. 4, pp. 685–695, Jul. 2020.
[33] M. J. Pazzani and D. Billsus, *Content-Based Recommendation Systems*. Berlin, Germany: Springer, 2007, pp. 325–341.
[34] M. Oppermann, R. Kincaid, and T. Munzner, "VizCommender: Computing text-based similarity in visualization repositories for content-based recommendations," *IEEE Trans. Vis. Comput. Graphics*, vol. 27, no. 2, pp. 495–505, Feb. 2021.
[35] S. Mahesh Kumar and R. Om Prakash, "Knowledge-based recommendation system for online business using web usage mining," in *Advances in Intelligent Systems and Computing*. Singapore: Springer, 2021, pp. 293–300.
[36] R. Burke, *Hybrid Web Recommender Systems*. Berlin, Germany: Springer, 2007, pp. 377–408.
[37] J. Wang, A. P. d. Vries, and M. J. T. Reinders, "Unifying user-based and item-based collaborative filtering approaches by similarity fusion," in *Proc. 29th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Aug. 2006, pp. 501–508.
[38] P. Valdiviezo-Diaz, F. Ortega, E. Cobos, and R. Lara-Cabrera, "A collaborative filtering approach based on Naïve Bayes classifier," *IEEE Access*, vol. 7, pp. 108581–108592, 2019.
[39] N. Shrivastava and S. Gupta, "Analysis on item-based and user-based collaborative filtering for movie recommendation system," in *Proc. 5th Int. Conf. Electr., Electron., Commun., Comput. Technol. Optim. Techn. (ICEECCOT)*, Dec. 2021, pp. 654–656.
[40] Y. Ren, Z. He, and T. Han, "Research on optimal design of online education course recommendation system based on hybrid recommendation algorithm," in *Proc. 2nd Int. Conf. Big Data Informatization Educ. (ICBDIE)*, Apr. 2021, pp. 461–464.
[41] D. F. Murad, R. Hassan, Y. Heryadi, B. D. Wijanarko, and Titan, "Recommendation system based on recognition of prior learning to support curriculum design in online higher education," in *Proc. Int. Conf. Inf. Manage. Technol. (ICIMTech)*, vol. 1, Aug. 2021, pp. 413–417.
[42] L. Salau, M. Hamada, R. Prasad, M. Hassan, A. Mahendran, and Y. Watanobe, "State-of-the-art survey on deep learning-based recommender systems for E-learning," *Appl. Sci.*, vol. 12, no. 23, p. 11996, Nov. 2022.
[43] F. L. da Silva, B. K. Slodkowski, K. K. A. da Silva, and S. C. Cazella, "A systematic literature review on educational recommender systems for teaching and learning: Research trends, limitations and opportunities," *Educ. Inf. Technol.*, vol. 28, no. 3, pp. 3289–3328, Mar. 2023.
[44] A. Khalid, K. Lundqvist, and A. Yates, "A literature review of implemented recommendation techniques used in massive open online courses," *Expert Syst. Appl.*, vol. 187, Jan. 2022, Art. no. 115926.
[45] Y. Shen, H. Li, and Z. Liao, "Online education course recommendation algorithm based on path factors," in *Proc. IEEE 5th Int. Conf. Inf. Syst. Comput. Aided Educ. (ICISCAE)*, Sep. 2022, pp. 257–260.
[46] L. Zhu, Y. Liu, X. Hei, Y. Wang, H. Meng, J. Jiao, and L. Pan, "A study on exercise recommendation method using knowledge graph for computer network course," in *Proc. Int. Conf. Netw. Netw. Appl. (NaNA)*, Dec. 2020, pp. 436–442.
[47] R. Yoshimura, K. Sakamoto, H. Washizaki, and Y. Fukazawa, "Recommendation system providing similar problems instead of model answers to programming assignments," in *Proc. IEEE 5th Eurasian Conf. Educ. Innov. (ECEI)*, Feb. 2022, pp. 229–232.
[48] X. Yu and W. Chen, "Research on three-layer collaborative filtering recommendation for online judge," in *Proc. 7th Int. Green Sustain. Comput. Conf. (IGSC)*, Nov. 2016, pp. 1–4.
[49] R. Yera and L. Martínez, "A recommendation approach for programming online judges supported by data preprocessing techniques," *Appl. Intell.*, vol. 47, no. 2, pp. 277–290, Sep. 2017.
[50] R. Y. Toledo, Y. C. Mota, and L. Martínez, "A recommender system for programming online judges using fuzzy information modeling," *Informatics*, vol. 5, no. 2, p. 17, Apr. 2018.
[51] R. Duan, C. Jiang, and H. K. Jain, "Combining review-based collaborative filtering and matrix factorization: A solution to rating's sparsity problem," *Decis. Support Syst.*, vol. 156, May 2022, Art. no. 113748.
[52] W. X. Zhao, W. Zhang, Y. He, X. Xie, and J.-R. Wen, "Automatically learning topics and difficulty levels of problems in online judge systems," *ACM Trans. Inf. Syst.*, vol. 36, no. 3, pp. 1–33, Mar. 2018.
[53] Y. Watanobe, M. M. Rahman, T. Matsumoto, U. K. Rage, and P. Ravikumar, "Online judge system: Requirements, architecture, and experiences," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 32, no. 6, pp. 917–946, Jun. 2022.
[54] C. M. Intisar and Y. Watanobe, "Classification of online judge programmers based on rule extraction from self organizing feature map," in *Proc. 9th Int. Conf. Awareness Sci. Technol. (iCAST)*, Sep. 2018, pp. 313–318.
[55] C. M. Intisar and Y. Watanobe, "Cluster analysis to estimate the difficulty of programming problems," in *Proc. 3rd Int. Conf. Appl. Inf. Technol.*, New York, NY, USA, Nov. 2018, pp. 23–28.
[56] M. Chaudhry, I. Shafi, M. Mahnoor, D. L. R. Vargas, E. B. Thompson, and I. Ashraf, "A systematic literature review on identifying patterns using unsupervised clustering algorithms: A data mining perspective," *Symmetry*, vol. 15, no. 9, p. 1679, Aug. 2023.
[57] K. P. Sinaga and M.-S. Yang, "Unsupervised K-means clustering algorithm," *IEEE Access*, vol. 8, pp. 80716–80727, 2020.
[58] J. M. John, O. Shobayo, and B. Ogunleye, "An exploration of clustering algorithms for customer segmentation in the U.K. retail market," *Analytics*, vol. 2, no. 4, pp. 809–823, Oct. 2023.

[59] F. Nie, Z. Li, R. Wang, and X. Li, "An effective and efficient algorithm for K-means clustering with new formulation," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 3433–3443, Apr. 2023.
[60] G. S. K. Priya, G. Pillai, A. Jana, S. Bandyopadhyay, T. Crosbie, and D. A. Ghanem, "An analytic hierarchy process (AHP) framework for feature evaluation of smart electricity meters in India," in *Proc. 56th Int. Universities Power Eng. Conf. (UPEC)*, Aug. 2021, pp. 1–5.
[61] W. C. Wedley, "Combining qualitative and quantitative factors—An analytic hierarchy approach," *Socio-Economic Planning Sci.*, vol. 24, no. 1, pp. 57–64, Jan. 1990.
[62] N. A. Azhar, N. A. Mohamed Radzi, I. S. Mustafa, K. H. M. Azmi, F. S. Samidi, I. T. Zulkifli, F. Abdullah, M. Z. Jamaludin, A. Ismail, and A. M. Zainal, "Selecting communication technologies for an electrical substation based on the AHP," *IEEE Access*, vol. 11, pp. 110724–110735, 2023.
[63] M. Ibrahim, I. S. Bajwa, N. Sarwar, F. Hajjej, and H. A. Sakr, "An intelligent hybrid neural collaborative filtering approach for true recommendations," *IEEE Access*, vol. 11, pp. 64831–64849, 2023.
[64] J. Hu, B. Hooi, S. Qian, Q. Fang, and C. Xu, "MGDCF: Distance learning via Markov graph diffusion for neural collaborative filtering," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 7, pp. 3281–3296, Jul. 2024.
[65] H. Xia, Y. Luo, and Y. Liu, "Attention neural collaboration filtering based on GRU for recommender systems," *Complex Intell. Syst.*, vol. 7, no. 3, pp. 1367–1379, Jun. 2021.
[66] H. Morise, K. Atarashi, S. Oyama, and M. Kurihara, "Neural collaborative filtering with multicriteria evaluation data," *Appl. Soft Comput.*, vol. 119, Apr. 2022, Art. no. 108548.
[67] R. A. El-Deen Ahmed, M. Fernández-Veiga, and M. Gawich, "Neural collaborative filtering with ontologies for integrated recommendation systems," *Sensors*, vol. 22, no. 2, p. 700, Jan. 2022.
[68] S. Peng, W. Han, and G. Jia, "Pearson correlation and transfer entropy in the Chinese stock market with time delay," *Data Sci. Manage.*, vol. 5, no. 3, pp. 117–123, Sep. 2022.
[69] A. N. Ngaffo, W. E. Ayeb, and Z. Choukair, "A Bayesian inference based hybrid recommender system," *IEEE Access*, vol. 8, pp. 101682–101701, 2020.
[70] T. T. Tran, V. Snasel, and L. T. Nguyen, "Combining social relations and interaction data in recommender system with graph convolution collaborative filtering," *IEEE Access*, vol. 11, pp. 139759–139770, 2023.
[71] M. S. Nogueira, J. I. D. Pinheiro, F. Assis, D. S. Menasché, P. Sermpezis, and T. Spyropoulos, "When should recommenders account for low QoS?" *IEEE Access*, vol. 11, pp. 132014–132036, 2023.
[72] B. Jeong and K. J. Lee, "NLP-based recommendation approach for diverse service generation," *IEEE Access*, vol. 12, pp. 14260–14274, 2024.
[73] B. Cao, Y. Zhang, J. Zhao, X. Liu, L. Skonieczny, and Z. Lv, "Recommendation based on large-scale many-objective optimization for the intelligent Internet of Things system," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 15030–15038, Aug. 2022.

DANIEL M. MUEPU (Graduate Student Member, IEEE) received the B.Sc. degree in information system design from the University of Kinshasa, Kinshasa, Democratic Republic of the Congo, in 2014, and the M.Sc. degree from the Graduate School of Computer Science and Engineering, Department of Computer and Information Systems, Aizuwakamatsu, Fukushima, Japan, in 2022. He is currently pursuing the Ph.D. degree with the Database Systems Laboratory, Department of Computer and Information Systems, The University of Aizu, Aizuwakamatsu. He has been a Junior Lecturer with the University of Kinshasa, since 2014. His research interests include machine learning, deep learning, recommender systems, data visualization, educational data mining, and machine learning applications in programming.

YUTAKA WATANOBE (Member, IEEE) received the M.S. and Ph.D. degrees from The University of Aizu, Japan, in 2004 and 2007, respectively. He was a Research Fellow with Japan Society for the Promotion of Science (JSPS), The University of Aizu, in 2007. He is currently a Senior Associate Professor with the School of Computer Science and Engineering, The University of Aizu. He is also the Director of i-SOMET and a Developer of Aizu Online Judge (AOJ) systems. His research interests include intelligent software, programming environments, smart learning, machine learning, data mining, cloud robotics, and visual languages. He is a member of IPSJ.