Recommending the Right Activities based on the Needs of each Student

Elias de Oliveira*, Márcia Gonçalves Oliveira and Patrick Marques Ciarelli
Programa de Pós-Graduação em Informática, Universidade Federal do Espírito Santo, Vitória, Brazil

Keywords: Activities Recommendation, Computers Programming, e-Learning, *k*NN Algorithm.

Abstract: Today more than ever, personalization is the most important feature that a system needs to exhibit. The goal of many online systems, which are available in many areas, is to address the needs or desires of each individual user. To achieve this goal, these systems need to rely on a good recommendation system. Hence, recommendation systems must work under the assumption that an individual's need could also be applied to someone else who has similar desires, tastes, and/or necessities. In this paper, we took this assumption into account and applied it to the context of learning computer programming. As a result, we present a system that recommends additional activities to students according to their individual needs. An additional assumption that is used is that a prompt reply and tailored guidance at each step of the learning process improves an individual's chances of success. We propose the use of the *k*NN algorithm to recommend activities that are as similar as possible to those that an expert would assign. The results are promising because we were able to mimic the human assessment decisions 90.0% of the time.

1 INTRODUCTION

We are overwhelmed by the amount of information that we need to process on a daily basis (Bawden and Robinson, 2009) to select the information that is relevant. Currently, one can find, either physically or online, a large number of items in numerous areas that might be of interest. Browsing all of these items and choosing one or some of these is sometimes stressful. To mitigate this difficult problem, an increasing number of studies have been conducted in the area of recommendation systems to help the user select those items that may be suitable.

A recommendation system is a program that analyses the behaviors and characteristics of users and attempts to recommend actions or items that could be useful to those users (Koren et al., 2009). Recommendation systems can be used in many different areas of knowledge to learn what users' interests are and to make recommendations accordingly (Herlocker et al., 1999). For instance, in e-commerce, recommendation systems can recognize customer profiles from their preferences or their buying habits and recommend products according to their interests (Linden et al., 2003). In medical systems, patient profiles can be associated with the presence of different symptoms, and medications can be recommended based on the symptom combinations (Meisamshabanpoor and Mahdavi, 2012; Chen et al., 2012). Similarly, in the area of e-learning, recommendation systems can map profiles based on the performances of students in different evaluation variables and offer them personalized instruction (Baylari and Montazer, 2009).

In the e-learning area, teaching computer programming is one of the modules that are considered to be difficult by many, particularly the students themselves. This type of knowledge is considered complex because it requires the combination of a set of cognitive skills and extensive practice of certain activities to achieve mastery (Pea and Kurland, 1984). To be successful and to encourage learning, the students must be well monitored on their practice of programming. Therefore, the goal of our recommendation system is to recommend the most appropriate activities for students such that they can improve their performances and reduce their difficulty as they learn programming.

To identify and automatically recommend activities in accordance with the evaluation variables that need improvement, we use a *k*NN (nearest neighbor) strategy (Soucy and Mineau, 2001). This is a baseline type of algorithm for recommendation systems (Tsai and Hung, 2012).

The *k*NN algorithm is a typical technique of col-

*The first author would like to thanks CAPES for the research grant 56128-12-2. And the second author would like as well to thank the FAPES/FUNCITEC foundation for their financial support for the development of this research under the project number 001/2009.

de Oliveira E., Gonçalves Oliveira M. and Marques Ciarelli P..
Recommending the Right Activities based on the Needs of each Student.
DOI: 10.5220/0004549501830190
In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval and the International Conference on Knowledge Management and Information Sharing* (KDIR-2013), pages 183-190
ISBN: 978-989-8565-75-4

laborative filtering recommendation systems because it makes recommendations based on the following three steps: building user profiles from user preferences, discovering user behaviors similar to the profile, and recommending the top-N preferred items from the nearest neighbors (Park et al., 2012).

When the implementation of our strategy was applied to our dataset, which is composed of 1,784 samples, the results showed that our strategy is effective for the recommendation of the most appropriate activities for different student profiles. The experiments indicated that our algorithm can mimic the instructor's assessment decisions most of the time.

This work is organized as follows. We present the problem and its context in Section 2. In Section 3, some related works are briefly reviewed. We discuss the results of an analogous problem tackled by a very simple strategy (kNN, when $k = 1$) in Section 4. This simple strategy will be compared against the more general kNN approach (kNN, when $k > 1$), and the first technique will be used as a lower bound in this work. In Section 5, we describe how the experiments were performed and the results obtained in the activities recommendation task. The conclusions are then presented in Section 6.

2 THE PROBLEM DESCRIPTION

In a learning environment, a recommendation system will be of great use if it can constantly monitor the coursework performances of the students. By analyzing their performances, this system could then recommend a tailored set of activities for the improvement of the learning performance of each individual student. Typically, a recommendation system uses information from other neighboring users to predict the choices that an active user would make. In the learning context, this strategy is analogous to the use of a set of neighboring students' profiles to suggest a range of activities to the target student, *i.e.*, activities that could change the target student's performance in terms of his/her learning progress. In other words, in this paper, we discuss the recommendation portion of the system (see Section 5) using our dataset, which structural features are similar to those of *MovieLens*, which is a traditional dataset used for movie recommendations (Herlocker et al., 1999), as discussed in Section 4.

In a more formal manner, consider D the domain of profiles and $\Omega = \{d_1, d_2, \ldots, d_{|D|}\}$ an initial corpus of profiles to which extra activities were previously manually assigned by an expert. A profile $d_j = (x_1, x_2, x_3, ..., x_{|d_j|})(\forall j = 1, 2, 3, \ldots |\Omega|)$ is composed of a set of $|d_j|$ assessment variables $v_q \geq 0$ $(\forall q = 1, 2, 3, \ldots, |d_j|)$. Each profile d_j of Ω is previously associated with a subset of activities $c_i \in C$ $(\forall i = 1, 2, \ldots, |C|)$. A recommendation system of activities implements a function $f : D \times C \rightarrow R$ that returns a value for each pair $(d_j, c_i) \in D \times C$, which indicates that the test profile d_j should be assigned activity $c_i \in A_p$, where $\cup_{p=1}^{m} A_p \subseteq C$. The function of the actual value $f(.,.)$ can be transformed into a ranking function $r(.,.)$, which is a one-to-one mapping onto $1, 2, \ldots, |C|$ such that $f(d_j, c_1) > f(d_j, c_2)$. This results in $r(d_j, c_1) < r(d_j, c_2)$. If A_p is the set of appropriate activities for test profile d_j, a good recommendation system should organize the activities for d_j within A_p over those that are not in A_p.

In this work, a formalization of the problem was applied as follows: D is the domain of profiles that are obtained from the programming activities of the students at

. The activities $c_i \in C$ are assumed to be related to more than one type of programming language module contents. Within profile d_j, each value v_q is the performance of a student on a cognitive q item within a programming activity. For example, item q can represent an understanding indicator of a programming concept or the proper use of an arithmetic, logical, or relational operator.

3 RELATED WORKS

Recommendation systems can be categorized according to their goal. For instance, in the e-commerce field, the goal is to finalize the business transactions, assist consumers in purchasing even more products, encourage consumers to buy other products, and/or conquer the loyalty of consumers (Drachsler et al., 2009). A slightly different goal is found in the educational context. In this case, the goal of recommendation systems for *Technology Enhanced Learning* (TEL) is to assist the students on their learning activities either by providing tips or actually suggesting a variety of activities to help them improve their learning experience. In these cases, the objective of the system was to support the development of the student's skills (Drachsler et al., 2009).

Other systems tackle the problem of finding alternative paths through learning sources (Manouselis et al., 2011). The Protus system is a proposed hybrid recommendation model that was developed to recommend a sequence or a path to be followed by selected sources of learning (Klašnja-Milićević et al., 2011). Protus recognizes student profiles through their habits and learning styles. From these profiles, the system

identifies clusters of students, discovers patterns, and recommends a sequence of activities based on frequent sequences of learning and perceived patterns in accordance with the profiles of the learners (Klašnja-Milićević et al., 2011). In this recommendation strategy, the sequence of suggested activities is based on a browsing history of the learner through the contents of a course. As a result, the system can find the habits and preferences of the learner but might not necessarily help their in the learning process. Thus, the Protus system could also assess whether the sequence of activities that is recommended to the learner does represent the most necessary activities that could improve the student's learning.

With respect to the algorithms used in this area, the *k*NN algorithm is a typical technique of collaborative filtering recommendation systems because it makes recommendations based on the following three steps: building user profiles from user preferences, discovering user behaviors similar to a profile, and recommending the top-*N* preferred items from the nearest neighbors (Park et al., 2012). The neighborhood selection consists of three main tasks: profile selection, profile matching, and best profile collection (Ujjin and Bentley, 2002). After profile selection, the profile matching process computes the distance or similarity between the selected profiles and the active user's profile using a distance function (Sarwar et al., 2001; Ujjin and Bentley, 2002).

Following these steps, as described by (Baylari and Montazer, 2009), we used the performance of students to compose multidimensional profiles in order to recognize learning difficulties that are characterized by the students' performances on different evaluation variables. We can then predict the evaluation variables of a student's performance profile based on the performances of other students on the same variables (Sarwar et al., 2001). This prediction may be based on the similarities between the students' profiles (Manouselis et al., 2010) or on the similarities between items, *i.e.*, between the values of the assessment variables (Sarwar et al., 2001; Lops et al., 2011). In this paper, we used the user similarity approach based on profiles. To develop a student's skills, as was noted by (Drachsler et al., 2009), we then recommend the most appropriate activities according to the values of the performance evaluation variables associated with the target student's profile (Tsai and Hung, 2012).

In the next section, we show some of the results of our strategy using the *MovieLens* dataset to compare and discuss the similarities of both problems.

4 RECOMMENDATION OF MOVIES

In general, a recommendation system considers that one's choices could also be applied to someone else with similar tastes. Therefore, one basic approach is to use a set of users' choices to make recommendations to the target user, which is also named the active user. The set of users with similar tastes will be called neighbors of the target user, and the neighborhood is usually found by calculating a threshold similarity value between the target user and the other users in the dataset (Herlocker et al., 1999).

In the area of movie recommendations, a very well-known dataset is included in *MovieLens*, which is provided by the GroupLens research group[2] (Park et al., 2012). Within this dataset, one can find users' profiles, their choices, and their actual ratings of movies. It is not necessary to mention that choosing movies is a very subjective type of decision making. Thus, the challenge in this area is to devise a system that recommends the right movies to a person based on their characteristics, their previous choices, and other knowledge available in their profile. Usually, this is achieved through a comparison with the profiles of similar users. In other words, the system attempts to find a user in the database who is very similar to the target user by searching for another user whose profile exhibits the most similar choices (Ujjin and Bentley, 2002).

The *MovieLens* dataset utilizes 17 features to characterize movies genres: (1) action, (2) adventure, (3) animation, (4) children, (5) comedy, (6) crime, (7) documentary, (8) drama, (9) fantasy, (10) film-noir, (11) horror, (12) musical, (13) mystery, (14) romance, (15) sci-fi, (16) thriller, (17) war, and western. An additional 4 features are used to profile the user: (1) movie rating, (2) age, (3) gender, and (4) occupation

(Herlocker et al., 1999) extensively evaluated a variety of features to analyze their effect on the accuracy of the system. One of their findings was that the number of neighbors used to predict the active user's rate affects the results. Based on their work, we propose the use of the *k*NN algorithm (Hao et al., 2007), and briefly discuss two situations. In the first situation, we choose only one neighbor (the nearest one), *i.e.* $k = 1$, to predict the outcome of the active user. This will be our lower-bound result for the additional experiments. In the second case, we choose a better neighborhood using more neighbors to improve the results, *i.e.* $k = 5$.

[2]http://www.movielens.umn.edu/

4.1 Recommending Movies based on the Nearest Neighbor

In this first approach, similarly to (Sarwar et al., 2001), we are interested in analyzing the accuracy of the results if we base our movie recommendations to the target user solely on the nearest neighbor.

We want to investigate the degree of accuracy of the lower-bound result. In this case, only the nearest neighbor user was used to determine the recommendation. Therefore, let $T_n = \{R_{n1}, R_{n2}, \ldots, R_{ni}, \ldots, R_{np}\}$ be the set of items rated by the nearest neighbor and $T_a = \{\ldots, R_{ai}, \ldots\}$ be that set of items rated by the active user. Note that each user can give different rates to the same movie. However, we calculate the predicted rate only when both the neighbor and the active user rated an item, e.g. $R_{n,i}$, and $R_{a,i}$. In this case, the predicted rate that the active user would use for this item is determined using the following equation: $\widehat{R_{ai}} = R_{ni} \times \phi$, where, in the case of the experiments included in Sections 4.1 and 4.2, we use the cosine similarity (Baeza-Yates and Ribeiro-Neto, 2011) between the active user and the nearest neighbor for the value of ϕ. The difference between the actual rate and the predicted rate will be computed as an error: $\text{MAE}_a = e_a = \frac{\sum_{i=1}^{N} |R_{ai} - \widehat{R_{ai}}|}{N}$.

In this equation, N is the total number of items that both the nearest neighbor and the active user rated. Thus, N is typically markedly less than the total number of items within the dataset. The results of this approach are shown in Table 1.

These results show that the average error is 0.9161. As shown in Figure 1(a), half of the error values are either above or below 0.91.

Table 1: Descriptive analysis of the error predictions.

Min.	Median	Mean	Max.	SD
0.0204	0.9041	0.9161	2.2631	0.2697

The first column in Table 1 (*Min.*) shows the minimum MAE error value obtained using this strategy. The second column (*Median*) shows the middle (median) value among the list of error values. The (*Mean*) column show the calculated mean value of the set of errors. The maximum error value is shown in the column titled (*Max.*), whereas the standard deviation is shown in the last column (*SD*).

In this case, the standard deviation is high at 0.9161 ± 0.2697. Due to this level of error, we may conclude that, in the worst case, we are missing the exact prediction match by, on average, 1 point over the N items rated by the pair of users (active user and nearest neighbor).

Table 2: Descriptive analysis of the error predictions.

Min.	Median	Mean	Max.	SD
0.0060	0.0316	0.0521	0.7343	0.0546

Based on the quadratic complexity nature of the procedure used to find the nearest neighbor, we argue that this is the simplest and the fastest strategy for this type of problem.

4.2 Recommending Movies based on the Neighborhood

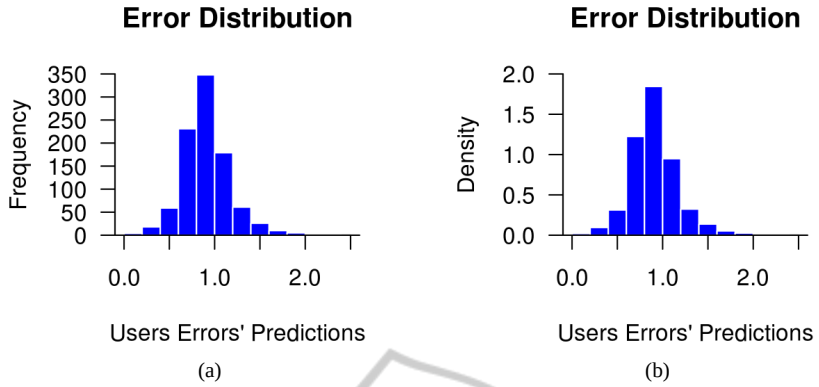
In this second approach, we exert a greater effort to find better results using the neighborhood to predict the ratings of the active user. As mentioned previously, the size and the quality of the neighborhood has a great impact on the predicted results (Herlocker et al., 1999). However, in this section, we will show only the results of a neighborhood constructed of the five nearest neighbors to the active user.

As shown in Figure 2, the results were improved by using a neighborhood for the prediction of the ratings. From the descriptive statistics, we found that the error decrease, on average, to 0.0521 compared with the previous results. Moreover, the results in Figure 2(a) demonstrate the new median value is 0.031. In summary, these results assert what was already noted by other authors in the literature on simple strategies. Although other strategies exist and have been proposed by other researchers, the strategy that used a combination of techniques yielded better results (Tsai and Hung, 2012).

5 RECOMMENDING ACTIVITIES

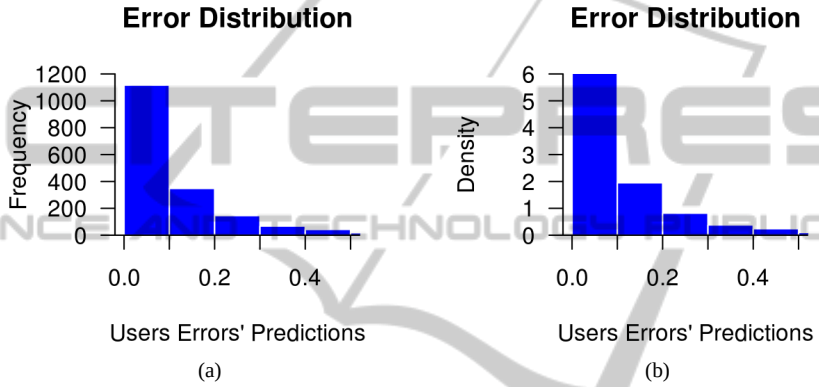
In our recommendation system, we follow the same strategy used with the *MovieLens* dataset to predict the user's ratings and recommend movies (Herlocker et al., 1999). The user-based collaborative filtering recommendation system was used to suggest activities for the students of a programming course. In this case, we predict the performance of a student in an assessment variable based on the performances of similar students in this same variable. After a value has been predicted for the assessment variable, we assessed whether this value is an indicator of learning success or a deficiency. If this value indicates a learning deficiency, activities related to this variable are recommended.

Through the prediction, we can anticipate a student's performance in different assessment variables to recognize their skills and difficulties in program-



(a) (b)

Figure 1: Error distributions found using the nearest neighbor approach



(a) (b)

Figure 2: The neighborhood approach.

ming practice and to evaluate their learning conditions for the development of new computer programs with various levels of complexity. Using the system to determine the learning states of the students through their predicted performances, we can improve the learning process of these students by providing them recommendations of specific activities according to their learning needs. The process of predicting the performances and recommending activities proposed in this paper uses the following steps (Herlocker et al., 1999): 1. Generate a dataset representing the performance of students through different evaluation variables; 2. Compute the similarity between the profiles of different students; 3. Predict the values of the evaluation variables of a student based on the profiles of the nearest students; 4. Recommend activities according to the values of the predicted variables; 5. Evaluate the results of the prediction. These steps are presented in detail in the following subsections.

5.1 The *ds-FAR* Dataset

For the prediction of a student's performance and the recommendation of activities, we generated a dataset denoted the *Formative Assessment Recommendation* (*ds-FAR*) that is composed of computer programs written in the programming language that were developed by students in a programming course. These programs were mapped to assessment variables representing the use of keywords, operations, commands, symbols, and structures of the *C* programming language.

The *ds-FAR* dataset consists of 1,784 samples of programs in the *C* programming language mapped to 37 assessment variables. These programs were developed by approximately 50 students for 39 activities proposed by an instructor. Unlike discrete ratings, *i.e.*, the values of the *MovieLens* dataset were limited between the values 0 and 5, the values $R_{jq} \geq 0$. Mapped from a profile d_j to the assessment variable v_q are continuous.

The programming activities within *ds-FAR* were divided into seven coursework activities and a final exam. Figure 4 shows an example of how three student profiles were represented by these assessment variables.

In Figure 4, the three profile states $A1$, $A2$ and $A3$ are represented by the performances of students in a programming activity that students solve by writing a computer program in the *C* programming language.

Student	v_1	v_2	v_3	v_4	v_5	v_6
A1	?	R_{12}	?	R_{12}	R_{12}	R_{12}
A2	R_{21}	R_{22}	R_{23}	R_{12}	R_{12}	-
A3	-	-	R_{33}	-	-	R_{36}
A4	R_{41}	R_{42}	R_{43}	R_{44}	R_{45}	-

Figure 3: Selection of predictors for the assessment variables.

Each written program is mapped to the assessment variables, and each variable v_q represents the frequency of occurrence of a reserved word, an operator in the *C* programming language, or an execution indicator of the compilation of the program and its correct execution. The values for each assessment variable v_q of a student profile were calculated by dividing the value of the variable by its corresponding value in the model solution chosen by the instructor. The comments and strings found in the programs written by the students were not considered in the assessment.

If the performance of a student in an assessment variable has a value less than 0.7, we assume that the student has difficulties using the *C* programming concept associated with the variable.

Similarly, if a student's performance on any of the assessment variables is above 1, s/he likely has some difficulties associated with the efficient use of the controls, structures, and operations of the *C* programming language that are represented by these variables. In other words, this particular student developed a computer program using more structures and operations than is required for resolving the given task.

In summary, the *ds-FAR* is a representative dataset for the prediction of student performances and for activities recommendations because it has many samples, it exhibits a great variability in the solutions developed by the students, and it contains a wide range of values for the different assessment variables.

5.2 Procedures

To predict the student performances, we utilized the traditional neighborhood-based recommendation method (see Section 4) using three steps (Herlocker et al., 1999): 1. Assign weights to the assessment variables of the student profiles based on their similarity with the active user; 2. Select a subset of student profiles to be predictors of an active user; 3. Normalize the values of the assessment variables and compute the prediction of a weighted combination of the nearest neighbors.

In Step 1, after we have represented the active user and the other students using multidimensional vectors, we calculated the similarity index between these profiles using the cosine similarity and formed a similarity matrix M_{sim}, where each of this matrix is the similarity between two profiles (d_m and d_p).

All of the variables of each student profile are then weighted by multiplying the similarity indexes between these profiles and d_a the profile of the active user.

In Step 2, using the *k*NN algorithm (Soucy and Mineau, 2001; Duda et al., 2001), we identified the *k* nearest neighbors to the active user d_a, *i.e.*, those students whose profiles exhibit higher rates of similarities with the profile of the active user. Thus, the neighbors that are selected as predictors to determine the value of assessment variable v_q of the active user are those with non-zero performance and those who have obtained better performances on the particular assessment variable of interest. In Figure 3, the hatched student profiles represent the selected predictors that will be used to predict the values and for variables R_{11} and R_{13}, respectively, of active user $A1$.

In Step 3, using a regression algorithm, we predict the performance of in assessment variable for the active user P_{aj}. $i_j \notin I_{da}$ is the assessment variable set in which the active user has already obtained performances. The prediction value is computed by the following equation: $P_{di} = \frac{\sum_{k=1}^{N}(s_{d,k} * R_{k,i})}{\sum_{k=1}^{N)}(|s_{d,k}|)}$

In this equation, N is the number of nearest neighbors to the active user d_a who have a value for assessment variable i. The value $s_{a,k}$ is the similarity index between the k neighbors and the active user. The value of $R_{k,i}$ is the performance of neighbor k in assessment variable i.

Finally, to analyze the predicted value for an assessment variable, we indicate what types of activities should be recommended to an active user.

When the predicted values in some assessment variables have values of less than 0.7, specific activities related to these concepts are recommended to help the student improve his/her performance in these assessment variables.

Similarly, if the predicted values in any of the assessment variables is higher than 1, activities are recommended to help the student more efficiently use the concepts of the programming language associated with the variable.

To evaluate the accuracy of the predictions obtained using our strategy with the *ds-FAR* dataset, we used the metric MAE, which was presented in Section 4 in the evaluation of the rating prediction using the *MovieLens* dataset.

5.3 Results

The results in Figure 5 demonstrate the efficacy of the user-based collaborative filtering recommendation

	assessment variables (attributes)																	
Students	compile	run	printf	include	main	return	+	-	*	/	%	++	--	=	float	int	char	scanf
A1	1	1	0.3	1	1	1	0	0	0	0.1	0	0	0	0	0	0.5	0	0
A2	1	1	2.4	1	1	1	0	1	0	0.2	0	0	0	0	0	1.5	0	0
A3	1	1	0.3	3	1	1	0	0	0	0	0	0	0	0	0	1	0	2

Students	>	<	!=	==	<=	>=	||	!	&&	if	else	else if	switch	case	break	default	do	while	for
A1	1	1	0	0	0	0	0	0	0	1	0.5	0	0	0	0	0	0	0	0
A2	1	1	0	5	0	0	0	1	0	2.5	2.5	0	1	1	1	1	0	1	0
A3	3	3	0	1	0	0	0	0	0	0.5	0.5	0	0	0	0	0	0	0	0

Figure 4: Student profiles mapped to assessment variables that represent the student's performances in different concepts of the *C* programming language.

strategy using the neighborhood-based method to predict student performances in programming activities.

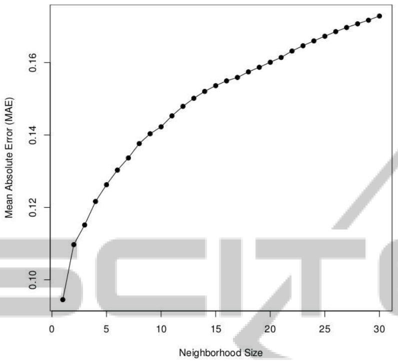


Figure 5: Results of the prediction of student performances using the *ds-FAR* dataset

As shown in Figure 5, the MAE values obtained for the *ds-FAR* dataset increase as the number of neighbors used in the evaluation is increased (Herlocker et al., 1999). However, if 30 neighbors are used for the prediction, the average MAE value for the *ds-FAR* dataset does not exceed 0.2, which indicates that the prediction results for the continuous assessment variables are satisfactory.

The maximum values of MAE for the *ds-FAR* dataset shown in Table 3 indicate poor predictions; however, these values could be improved if the values that are greater than 1 are normalized to a unique value, as performed in Step 2 of the experimental procedures, because any value greater than 1 represents an inefficient use of the programming instructions.

The results presented by (Herlocker et al., 1999), who analyzed the *MovieLens* dataset using different numbers of neighbors in the range of 0 to 100, revealed that the MAE error slightly oscillated between 0.69 and 0.71. As shown in Table 3, the results obtained for the *ds-FAR* dataset were well above those obtained using the *MovieLens* dataset.

Table 3: Descriptive analysis of the error predictions obtained with the *ds-FAR* dataset.

k	Min.	Median	Mean	Max.	SD
1	0.0000	0.0181	0.0945	7.2810	0.2692
30	0.0000	0.1146	0.1729	7.3600	0.2650

The superiority of the results obtained with the *ds-FAR* dataset compared with the *MovieLens* dataset might be the result of the difference in the level of sparsity between these datasets. The non-null items in the *ds-FAR* dataset constitute 38.8% of the total items, whereas the non-null items in the *MovieLens* dataset are 6.3%, which indicates that the *ds-FAR* dataset is less sparse than the *MovieLens* dataset.

According to (Sarwar et al., 2001), the nearest neighbor algorithms exhibit poor performance with large and sparse datasets. However, for datasets that are not as wide and sparse as the *ds-FAR* dataset, these algorithms can exhibit good performance, as indicated by our results.

6 CONCLUSIONS

In this paper, we proposed a strategy for profiling and then recommending activities to students with learning difficulties in computer programming. The personalized recommendations give each individual student specific suggestions to improve their learning experience by performing additional individualized activities.

The comparison of the results obtained with the recommendation system and that produced by an expert revealed that the system was able to imitate the human expert up to 90.0% of the times. This finding also showed that we can greatly reduce the effort of the instructor through the use of this approach. Moreover, the students would benefit the most because they would have additional prompt support throughout their learning process. In future work, we plan to apply this strategy to a larger audience and to more classes to further analyze the behavior and the quality of the algorithm.

REFERENCES

Baeza-Yates, R. and Ribeiro-Neto, B. (2011). *Modern Information Retrieval*. Addison-Wesley, New York, 2 edition.

Bawden, D. and Robinson, L. (2009). The Dark Side of Information: Overload, Anxiety and other Paradoxes and Pathologies. *J. Inf. Sci.*, 35(2):180–191.

Baylari, A. and Montazer, G. (2009). Design a Personalized e-Learning System Based on Item Response Theory

and Artificial Neural Network Approach. *Expert Systems with Applications*, 36(4):8013 – 8021.

Chen, R.-C., Huang, Y.-H., Bau, C.-T., and Chen, S.-M. (2012). A Recommendation System Based on Domain Ontology and SWRL for Anti-Diabetic Drugs Selection. *Expert Systems with Applications*, 39(4):3995 – 4006.

Drachsler, H., Hummel, H. G. K., and Koper, R. (2009). Identifying the Goal, User Model and Conditions of Recommender Systems for Formal and Informal Learning. *J. Digit. Inf.*, 10(2).

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification*. Wiley-Interscience, New York, 2nd edition.

Hao, X., Tao, X., Zhang, C., and Hu, Y. (2007). An Effective Method to Improve *k*NN Text Classifier. In *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, volume 1, pages 379–384.

Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. (1999). An Algorithmic Framework for Performing Collaborative Filtering. In *22nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 230–237, Berkeley, California, USA. ACM.

Klašnja-Milićević, A., Vesin, B., Ivanović, M., and Budimac, Z. (2011). E-Learning Personalization Based on Hybrid Recommendation Strategy and Learning Style Identification. *Computers & Education*, 56(3):885 – 899.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):30–37.

Linden, G., Smith, B., and York, J. (2003). Amazon.com Recommendations: Item-To-Item Collaborative Filtering. *Internet Computing, IEEE*, 7(1):76 – 80.

Lops, P., Gemmis, M., and Semeraro, G. (2011). Content-based Recommender Systems: State of the Art and Trends. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, pages 73–105. Springer US.

Manouselis, N., Drachsler, H., Vuorikari, R., Hummel, H., and Koper, R. (2011). Recommender Systems in Technology Enhanced Learning. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, pages 387–415. Springer US.

Manouselis, N., Vuorikari, R., and Van Assche, F. (2010). Collaborative Recommendation of e-Learning Resources: an Experimental Investigation. *Journal of Computer Assisted Learning*, 26(4):227–242.

Meisamshabanpoor and Mahdavi, M. (2012). Implementation of a Recommender System on Medical Recognition and Treatment. *International Journal of e-Education, e-Business, e-Management and e-Learning*, 2(4):315–318.

Park, D. H., Kim, H. K., Choi, I. Y., and Kim, J. K. (2012). A Literature Review and Classification of Recommender Systems Research. *Expert Systems with Applications*, 39(11):10059–10072.

Pea, R. D. and Kurland, D. (1984). On the Cognitive Effects of Learning Computer Programming. *New Ideas in Psychology*, 2(2):137 – 168.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-Based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA. ACM.

Soucy, P. and Mineau, G. W. (2001). A Simple KNN Algorithm for Text Categorization. In *ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 647–648, Washington, DC, USA. IEEE Computer Society.

Tsai, C.-F. and Hung, C. (2012). Cluster Ensembles in Collaborative Filtering Recommendation. *Applied Soft Computing*, 12(4):1417–1425.

Ujjin, S. and Bentley, P. (2002). Learning User Preferences Using Evolution. *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning, Singapore*.