Knowledge-Based Systems 37 (2013) 283–295

Contents lists available at SciVerse ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

The development of intuitive knowledge classifier and the modeling of domain dependent data

H. Tolga Kahraman [a,*], Seref Sagiroglu [b], Ilhami Colak [c]

[a] Karadeniz Technical University, Faculty of Technology, Software Engineering Department, Turkey
[b] Gazi University, Faculty of Engineering, Computer Engineering Department, Turkey
[c] Gazi University, Faculty of Technology, Electric Electronic Engineering Department, Turkey

ARTICLE INFO

Article history:
Received 13 February 2012
Received in revised form 6 August 2012
Accepted 11 August 2012
Available online 21 August 2012

Keywords:
Web-based user modeling
Domain independent object model
Intuitive knowledge classifier
Weight-tuning method
AEEC user modeling server

ABSTRACT

Creating an efficient user knowledge model is a crucial task for web-based adaptive learning environments in different domains. It is often a challenge to determine exactly what type of domain dependent data will be stored and how it will be evaluated by a user modeling system. The most important disadvantage of these models is that they classify the knowledge of users without taking into account the weight differences among the domain dependent data of users. For this purpose, both the probabilistic and the instance-based models have been developed and commonly used in the user modeling systems. In this study a powerful, efficient and simple 'Intuitive Knowledge Classifier' method is proposed and presented to model the domain dependent data of users. A domain independent object model, the user modeling approach and the weight-tuning method are combined with instance-based classification algorithm to improve classification performances of well-known the Bayes and the *k*-nearest neighbor-based methods. The proposed knowledge classifier intuitively explores the optimum weight values of students' features on their knowledge class first. Then it measures the distances among the students depending on their data and the values of weights. Finally, it uses the dissimilarities in the classification process to determine their knowledge class. The experimental studies have shown that the weighting of domain dependent data of students and combination of user modeling algorithms and population-based searching approach play an essential role in classifying performance of user modeling system. The proposed system improves the classification accuracy of instance-based user modeling approach for all distance metrics and different *k*-values.

1. Introduction

User modeling is one of the most powerful mechanisms for the web-based adaptive applications. Researchers use students'/users' model to customize subjects according to their knowledge. The goal of a user modeling system is to provide enough or suitable knowledge for students/users. Thus, the adaptation of web content is the basic requirement of web-based adaptive applications. In this process, user models are used for content personalization, accessing the data easily and fast, and streamlining the applications more effective and efficient. The user models have been mostly created in web-based applications, especially for online learning environments.

Students' models are composed of static and dynamic data in a web-based adaptive learning environment. Static data represent information about the students such as username, password, and age. Dynamic data represent knowledge of students about the domain dependent data. Thus, the dynamic data in user model might be also called the user knowledge model [1]. The domain dependent data are about the students' educational activities such as visited pages, elapsed time in the concept/goal pages and exam performance in lesson pages, and keystrokes.

The behaviors or data of students are obtained from their interactions with web-environment by the user modeling system (UMS). UMS tracks students' learning activities, navigation paths and stores this data in their user model. This data are used as input data by a user modeling algorithm aimed at creating a consistent description of the model of students [1]. After that, the input data is used to create and to update user knowledge model.

The machine learning algorithms, the educational hypermedia object models and the user modeling approaches are crucial elements for a UMS. One difficulty key in UMS design process is the integration of these three basic elements with a powerful framework [2]. These elements play a decisive role in design process and performance of UMS. They are used to evaluate the stored data and classify the students' knowledge.

* Corresponding author.
E-mail addresses: htolgakahraman@yahoo.com (H.T. Kahraman), ss@gazi.edu.tr (S. Sagiroglu), icolak@gazi.edu.tr (I. Colak).

0950-7051/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.knosys.2012.08.009

One of the challenges is to decide which data should be stored in users' model. So, data selection is important for an efficient modeling of knowledge of users because it is seen as a crucial element of user knowledge and unique information in the personalization process. It directly affects the performance of UMS.

The other challenge is to decide which hypermedia object model is the most appropriate to form the application domain. Various object models have been used to form the domain dependent data of web-based adaptive learning environments [3–5].

A current challenge is to decide which learning algorithms are most appropriate for different types of user modeling tasks. The algorithm considerably affects the performance and reduces/increases the response time of UMS. For this purpose, the probabilistic and the instance-based user modeling algorithms have been developed and their effectiveness explored in web-based adaptive courses [5–8]. For example, the Bayes-based probabilistic classifier might obtain the best performance from a given training data, but it can be quite costly to apply. Computing of conditional probability distributions is quite cost for every hypothesis [7–10]. Therefore, new approaches and types of UMS will be needed in web-based learning courses. The most basic instance-based method is the *k*-nearest neighbor (*k*-NN) classifier. This classifier has two advantages: the data is processed by conceptually straightforward approach to approximating real-valued or discrete-valued target functions, and the training process consists of simply storing the collected training data [10]. The major drawbacks of these methods are the lack of data [11,12] and strongly application-dependency and lack of classification accuracy [11,13]. For example, students usually provide a small number of feedback data in the web-based courses while these models need a large number of training data.

To overcome the problems mentioned above, an Intuitive *k*-NN Knowledge Classifier (IKC) has been introduced, developed and tested in this study. The IKC was developed based on a hybrid genetic algorithm [14]. The IKC approach is classifier-independent and can be combined with different classification methods like the Bayesian classifier. One of most important features of this classifier is to create the knowledge model of a user for each learning object domain independently. IKC also discovers the relationships among user activities and learning objects. The importance of users' activities/features is determined in terms of a learning object in the discovering process of relationships. For this purpose, a domain-independent object model, a robust and an effective user modeling approach, an instance-based classification algorithm and a novel weight-tuning population based method were also combined in this study.

The steps of progresses followed in this study are as: firstly, the generic object model developed by Kahraman in the Adaptive Educational Electric Course (AEEC) [5] were used to form application domain (domain model) and knowledge model of students. The generic object model can easily be adapted to form dynamic and static models. Secondly, user modeling approach of web-based AEEC course was then adopted to select the students' data and determine user modeling strategy used in user modeling process. Thirdly, the probabilistic and the instance-based classification algorithms used to predict the class of user knowledge were adapted to the user modeling system of AEEC. Thus, an instance-based knowledge classifier algorithm was developed. After that, the instance-based knowledge classifier was extended to improve its classification performance. The algorithm was combined with a weight-tuning method and an intuitive searching approach. For this purpose, the genetic algorithm-based weight-tuning method was adopted to adjust the weights of students' domain dependent data. The proposed knowledge classifier has explored the best weights domain dependent data (features) of students on their knowledge class. A weight value in the interval [0–1] is assigned to each domain dependent data of student in weighting process. After that, IKC measures the distances among the students depending on the weight values and students' domain dependent data.

Finally, it uses the dissimilarities in the classification process to determine knowledge class of students. The proposed IKC is a hybrid classification algorithm transforming the features of objects to weight vector with combination *k*-NN classifier, population-based weight-tuning method and various distance metrics.

There are mainly five contributions of this study to literature which are:

(1) presenting a generic object model for modeling the interactions of users with system in web-based hypermedia environment practically tested in AEEC
(2) introducing a population-based weight-tuning method to explore the domain dependent data of students
(3) adopting a user modeling approach that compatible with different application domains
(4) increasing the classification accuracy of the well-known and commonly used probabilistic and instance-based user modeling algorithms
(5) presenting a practicable and easily to understand an intuitive knowledge classifier algorithm producing effective results to the user modeling literature.

The organization of the paper is as follows: Section 2 presents related study. Section 3 presents the domain independent object model, the architecture and features of AEEC user modeling system. Section 4 presents the details of probabilistic and instance-based knowledge classifiers. A probabilistic Bayes knowledge classifier algorithm developed and tested in AEEC is presented. After that, the development process of instance-based knowledge classifier depending on the domain-independent object model and AEEC user modeling approach are introduced in Section 4. It also presents the population-based a weight-tuning method to adjust the weight values of domain dependent data of students. The weight-tuning method is combined with the knowledge classifier algorithm to improve the performance of it. The results of the experimental study are presented in Section 5. The probabilistic and instance-based knowledge classification algorithms are tested and their performances are compared with each other. Section 6 concludes the paper and discusses future work.

2. Related study (user modeling approaches)

Adaptive Educational Hypermedia Systems (AEHSs) were developed under the influence of studies in the area of adaptive hypermedia (AH), **c**omputer **a**ided **l**earning (CAL) and **i**ntelligent **t**utoring **s**ystems (ITS) [15–18]. The architectures and domain models of AEHSs were developed on the basis of the hypermedia reference models. Models such as, Dexter [15] the first reference model for hypertext systems, AHAM [16] presenting a basic reference model for hypermedia systems and SAHM [17] producing intelligent solutions for new generation web-based adaptive applications are the significant reference architectures in area of adaptive hypermedia. Designing the adaptive hypermedia environment independently from domain models is an important problem. According to Brusilovsky [19], the design of an adaptive hypermedia environment involves three sub-steps: i – structuring the hypermedia, ii – structuring the knowledge model and iii – the defining the relations between the hypermedia object model and knowledge model. The keys for adaptation are the application of domain model and the user knowledge model [6]. In INSPIRE [20,21], the application domain model is organized into three hierarchical levels as: learning goals, concepts and materials [20,21]. In

GIAS [6], the course model is structured in the goal layer, the topic layer and the resource layer. Goals are associated with a set of topics and the topics are associated with a set of resources. In LAOS [4], the domain model is composed of concept maps and linked concepts. The domain model represents learning resources and their characteristics. Domain model is organized into three hierarchical levels in the AEEC [5] such as learning goals, topics and concepts. Every layer has educational materials used as learning resources for objects. The goal objects can be associated with a set of goals, topics and materials. The topics can be associated with a set of topics, concepts and materials. Finally, the concepts can be associated with a set of concepts and materials.

A common feature of these models is that the course model of an adaptive application is divided into sub-layers. The types and features of these layers might be different. There is no a common object model used in as application-independent to structure the domain models of applications. However, there are a number of common objects such as goals and materials. Consequently, a generic object model should be designed. It is also needed to develop a domain-independent UMS.

The studies in ITS area are effective during creating the user model keys for personalization and adaptation. In last 10 years, the studies about user modeling as an independent research subject have become one step forward of ITS studies [18,22,23]. Personal features of users and knowledge about the application domain are presented in the user models,

One task of the user modeling systems is to track and store user actions. Another task is to evaluate and convert the stored data into useful information. In the evaluation process of user data, mostly rule-based approaches and machine learning algorithms especially for classification/prediction/clustering are used [5,8,18]. O'Mahony and Smyth [24] successfully described a supervised classification approach designed to identify and recommend the most helpful product reviews. They compared the performance of JRip, J48 and Naïve Bayes classification techniques using a range of features derived from hotel reviews. Reviewing instances/samples consisted of features derived from four distinct categories mined from individual reviews and the wider community reviewing activity. Prior to classification, each review was translated into a feature-based instance representation. One disadvantage of the approach is that all of the features have same effect in the decision-making process of classifiers. In the experimental studies, the AUROC scores (area under ROC curve) of classifiers did not exceed 82%. Weighting of the features might be increasing the AUROC scores of classifiers.

Virgilio et al. [25] developed a rule-based approach supporting the automatic adaptation of content delivery in Web Information Systems. The adaptation is accomplished automatically via production rules that it specifies how a configuration can be generated to meet the requirements of adaptation of a given profile.

Martins et al. [26] defined the user models as the characteristic in which the preferences and knowledge of user are presented by the system. The user models have the key role to enable the adaptive content and adaptive navigation paths. Bezold [27] also defined the user activities in the interactive systems as sequence of events and described a task model regarding these events. Bezold used 'probabilistic deterministic finite-state automata PDFA' in order to describe the user attitudes. The interaction history was converted into a vector set in the study. This vector set was used as an input parameter for the approach of 'first-order Markov chains' and so that next user activity was estimated. It was suggested a task-based on user modeling approach in order to form the mental model of a knowledge worker [28]. The main problems of task-based user modeling are that there is no standard conceptual model and no methodology developed for evaluating the methods used [28].

The reference studies [27–29] examined for task-based user modeling approach. The reference study [30], focused on producing recommendations for text-based resources. Therefore, it shaped the user models according to overlay model approach. First of all, the page contents that the user has visited are analyzed. After the analysis, the tags and keywords are removed from the content. In this approach, content and tag based approaches are combined. The similarity between the users is determined considering the pages visited by the users, tags and the keywords of these pages [30].

Pardos and Heffernan [22] highlighted that there was not enough study about estimating the initial knowledge of users and so they adopted knowledge tracking method. In this method, they have used multiple prior knowledge parameters 'initial knowledge parameter', 'learn rate', 'guess rate' and 'slip rate'. They have used a Bayesian Network in order to learn the parameters of the 'Prior Per User Model' and test its performance. As a result, it is much more effective in presenting the individualized users rather than standard knowledge tracing method.

The references [5,8,9,26] can be examined in order to gather information about the studies adopting the approaches of simple overlay model, concept-based overlay model, stereotypes, complex episodic student model, rule-based modeling during the user modeling process. Various committees are working on standardization of the user data in order user models to be interoperable among different systems [26].

Kahraman [5] and Martins et al. [26] reported that considering the historical data of the users during user modeling process makes a significant contribution to efficiencies of the models and the success during teaching/learning process while the users are registering to the system in AEEC, they declare their previous experiences. The users fill in an information form about whether they joined such a course before and if so whether they succeed or not. Depending on the previous experiences and knowledge of the users, the user modeling system picks the suitable one amongst various user modeling strategies. In this regard, the user modeling strategy in AEEC is adaptable [5].

In literature, although there are many approaches developed for modeling the user knowledge, it is seen that most of them are not used for real-world applications. The basic reasons of this problem turn out to be the dependency of the developed user modeling approaches to the application domain, their complexity, their processes, their focuses on the subject of theoretic modeling (inability of applicability) and also only few of them have been tested by real data sets [26]. According to the literature, each of the users' features is equally important to measure the distances among them [24–27,31,32]. In other words, UMSs assign equal weight values for every feature of user in knowledge modeling process. This is one of the main causes of misclassifications in knowledge modeling. Therefore, the importance or weight values of users' features on their knowledge class should be determined by the UMSs. The weighting of users' features will improve the classification results and estimations of UMSs. Consequently, a domain-independent object model, a classification method suitable for user modeling tasks and a flexible weight-tuning method should be combined with a powerful UMS framework to develop an effective UMS.

3. Generic object model and user modeling approach

In this section, the hypermedia object models commonly used in domain model design for hypermedia systems are introduced briefly and the generic object model and the user modeling approach of AEEC are also presented.

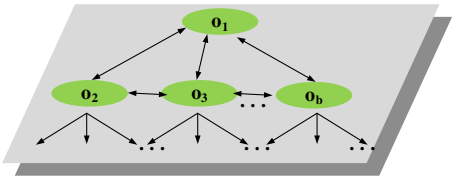

Fig. 1. Generic object model for web-based adaptive environments [5].

3.1. Generic object model

To represent the objects of a web-based adaptive application, a generic object model is given in Fig. 1. It is an extension of AEEC object model and MAID meta-model [3]. The goals, topics, concepts and other specific learning objects of an educational application might be defined domain-independently in this layer. Thanks to the generic object model, the hierarchical relations among the different types of learning objects might be easily defined. Most popular relations in adaptive learning environments are prerequisite connections among the learning objects representing the instructional requirement for one of the related learning objects [19].

According to the generic object model, a set of educational objects is represented with $\mathbf{O}$, $<o_1, o_2, o_3, \ldots, o_b>$ for an application domain. 'b' denotes the number of objects in domain model. The set of objects may include multiple types of objects such as goal object, topic object, and concept object. In the AEEC [5], the features of objects and their relationships: materials, prerequisites, and related objects are defined by the lecturer/designer in XML-based object description documents. Bidirectional relations among the objects are represented with '↔' arrows [5,18]. It supposes 'c' $\in \mathbf{N}$ and $\forall\ \mathbf{o_c} \in \mathbf{O}$. The individual features of $\mathbf{o_c}$ object are represented by the set of $\mathbf{Fo_c}$ $\langle fo_{c1}, fo_{c2}, fo_{c3}, \ldots, fo_{cn}\rangle$ in any web-based learning environment arranged in accordance with the generic object model. 'n' denotes the number of features in the set. The prerequisite objects for $\mathbf{o_c}$ are represented by the set of $\mathbf{Ro_c}$ $<ro_{c1}, ro_{c2}, ro_{c3}, \ldots, ro_{cv}>$, '$v$' denotes the number of related objects and every object in the set of $\mathbf{Ro_c} \in \mathbf{O}$. The individual and the relational features of an object $\mathbf{o_c}$ are represented in Table 1. According to the generic object model, the features and prerequisite relationships of learning objects should be defined to organize domain model of an application when designing web-based adaptive learning environment. In the following sub-section, the knowledge of a student for the object $\mathbf{o_c}$ will be diagnosed by UMS depending on the student's activities and data about the features and prerequisite objects of $\mathbf{o_c}$.

Depending on the object model of AEEC, every feature of educational objects might have one or more definitions. Some of the features are the object difficulty level, repetition number, questions, exams, and optimal time interval $[to_{c1}, to_{c2}]$. Instructors/lecturers can specify the real-values of features for each learning object such as $\mathbf{o_c}$. For example, instructors can define a measure of belief of the learner's understanding of the learning objects depending on the study time-interval. Some of the concrete features of learning objects in the application domain model of AEEC are given in Table 2 [5].

3.2. User modeling approach

The AEEC modeling approach results in obtaining the feature set, which will be used for classification purposes in the next part of the data analysis. Therefore, the AEEC modeling can be also viewed as a feature discovery approach.

Table 1
Representation of prerequisite objects and the individual features of $\mathbf{o_c}$ object.

Set of $\mathbf{Fo_c}$ (individual features of $\mathbf{o_c}$ object)				
fo_{c1}	fo_{c2}	fo_{c3}	$\cdots$	fo_{cn}
Set of $\mathbf{Ro_c}$ (prerequisite objects for $\mathbf{o_c}$)				
ro_{c1}	ro_{c2}	ro_{c3}	$\cdots$	ro_{cv}

3.2.1. The structure and features of user model

A user model is the main source of adaptation and personalization in web-pages. There is a strong relationship between the domain model and the knowledge model of users in web-based adaptive learning environments. The majority of UMSs use *overlay model approach* to create students' knowledge model [5,19]. The key principle of this approach for each learning object in application domain, individual student knowledge model stores some data used to predict students' current knowledge level (class) about learning objects. Fig. 2 shows the structure of user knowledge model. The domain dependent data of students corresponding to the learning objects in application domain is represented by real-valued features or knowledge items such as $\langle Umo_1, Umo_2, \ldots, Umo_b\rangle$. For example, the knowledge item $\langle Umo_c\rangle$ denotes the activities, the study time, the repetition number, and the knowledge level (class) of m*th* user about the learning object $\mathbf{o_c}$ and the prerequisite objects of it. Knowledge items $\langle Umo_1, Umo_2, \ldots, Umo_b\rangle$ are represented by sets of users' features: the sets of $\mathbf{Fu_{xx}}$ and $\mathbf{Ru_{xx}}$. Real-valued features in the sets of $\mathbf{Fu_{xx}}$ and $\mathbf{Ru_{xx}}$ constitute the domain dependent data of users. The users' features corresponding to the object $\mathbf{o_c}$ are given in Table 3 (the users' features about the object $\mathbf{o_c}$ also are represented by the $\langle Umo_c\rangle$).

The students' data or knowledge about the individual features of object $\mathbf{o_c}$ is represented by the set of $\mathbf{Fu_c}$ and the prerequisite objects (relational features) of object $\mathbf{o_c}$ are represented by the set of $\mathbf{Ru_c}$. Table 3 is structured according to Table 1.

It is obvious that considering users' features in the sets of $\mathbf{Fo_c}$ and $\mathbf{Ro_c}$ have same effect on their knowledge level (class) In other words, each of these features might have been different effect on the knowledge of students for a learning object. In this case, current challenges are to explore the effectiveness of the domain dependent data or features of students on their knowledge or to weight the features in the sets $\mathbf{Fu_c}$ and $\mathbf{Ru_c}$. The aim of weighting is to model the real-values of each feature in $\mathbf{Fu_c}$ $\langle fu_{c1}, fu_{c2}, fu_{c3}, \ldots, fu_{cn}\rangle$ and $\mathbf{Ru_c}$ $\langle ru_{c1}, ru_{c2}, ru_{c3}, \ldots, ru_{cv}\rangle$ on knowledge level or class of students. Classifying knowledge of users based on weighted feature sets eliminates the significant problems so that all features have the same effect. The weighted feature set is given in Table 4. The array of $\langle w_{1f}, w_{2f}, w_{3f}, \ldots, w_{nf}\rangle$ denotes the weights of features in a set of $\mathbf{Fu_c}$ and the array of $\langle w_{1r}, w_{2r}, w_{3r}, \ldots, w_{vr}\rangle$ also denotes the weights of features in a set of $\mathbf{Ru_c}$.

In order to express the details, it is better to explain the parameters. UMS classifies the knowledge levels (UNS) of users depending on the real-values of these features (see Table 5). There are five different features $\langle$STG, SCG, PEG, STR, LPR$\rangle$. STG, SCG and PEG are about the learning objects and the others about the prerequisite objects are used to classify the current knowledge of a user about

Table 2
Some of the concrete features of objects in application domain model of AEEC [5].

Individual features of object (corresponding to the set of $\mathbf{Fo_c}$)				The features of objects associated with object $\mathbf{o_c}$ (corresponding to the set of $\mathbf{Ro_c}$)		
Time interval	Repetition number	Difficulty level	Questions	Time interval	The knowledge level to be learned	Questions

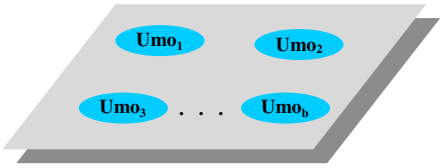

Fig. 2. The structure of user knowledge model [5].

Table 3
Users' features about learning object o_c.

$\boldsymbol{Fu_c}$ *(users' features about the set of* $\boldsymbol{Fo_c}$*)*				
fu_{c1}	fu_{c2}	fu_{c3}	⋯	fu_{cn}
$\boldsymbol{Ru_c}$ *(users' features about the set of* $\boldsymbol{Ro_c}$*)*				
ru_{c1}	ru_{c2}	ru_{c3}	⋯	ru_{cv}

Table 4
The representation of the weighted feature set.

$\mathbf{wFu_c}$				
$w_{1f} * fu_{c1}$	$w_{2f} * fu_{c2}$	$w_{3f} * fu_{c3}$	⋯	$w_{nf} * fu_{cn}$
$\mathbf{wRu_c}$				
$w_{1r} * ru_{c1}$	$w_{2r} * ru_{c2}$	$w_{3r} * ru_{c3}$	⋯	$w_{vr} * ru_{cv}$

Table 5
Some of concrete features of users about objects in application domain of AEEC [5].

Users' features about learning objects (corresponding to the $\boldsymbol{Fu_c}$*)*			
The degree of study time (STG)	The degree of repetition number (SCG)	The performance in exams (PEG)	The knowledge level (UNS)
Users' features about the prerequisite objects (corresponding to $\boldsymbol{Ru_c}$*)*			
The degree of study time (STR)	The knowledge level (Learning status) (LPR)		

the learning objects in AEEC. The definitions of users' features; the degree of study time (STG), the degree of repetition number (SCG) and the user performance in exams (PEG) for learning object $\mathbf{o_c}$, the degree of study time (STR) and the learning percentage (LPR) of users for prerequisite objects are given in Table 5. The current knowledge of students (UNS) is determined using real values of ⟨STG, SCG, PEG, STR, LPR⟩ as input parameters of the user modeling algorithm in AEEC. The proposed IKC determines the knowledge class of students considering both of the real values of students' features ⟨STG, SCG, PEG, STR, LPR⟩ and the weight values ⟨W_{STG}, W_{SCG}, W_{PEG}, W_{STR}, W_{LPR}⟩ in the modeling process. Thus, the weighting of user's features is an essential part of UMS in IKC approach.

3.2.2. The architecture of UMS

Fig. 3 shows the architecture of the proposed IKC. It is based on UMS architecture. The knowledge of users are created and updated in five steps (Steps 1–6) in AEEC. In the proposed IKC approach, there are six steps. The first step is to collect users' data. UMS tracks and collects the users' data such as learning activities/feedbacks/answers/navigation paths about the learning objects and prerequisite objects. Reading texts, solving problems/exercises/tests, navigation in the different pages of a learning environment are several examples of available online data. In the second step, the collected data is stored in the knowledge models of users. The stored data is then evaluated and converted to the useful data by the rule-based system in the third step. The useful data represents the real values of users' five different features in the sets of $\mathbf{Fu_c}$ ⟨STG, SCG, PEG⟩ and $\mathbf{Ru_c}$ ⟨STR, LPR⟩. These five steps are common in AEEC and IKC approaches. Please refer to [5,8,9] for detailed information about the rule-based system. In AEEC, the knowledge of users (UNS) for a learning object is classified depending on the real values of their features ⟨STG, SCG, PEG, STR, LPR⟩ about that object. In the proposed IKC approach, the features of users are weighted by the weight-tuning method (see fourth step in Fig. 3). After that, an optimum weight array is constructed in the form of [W_{STG}, W_{SCG}, W_{PEG}, W_{STR}, W_{LPR}]. The knowledge of users for a learning object is classified depending on the real values of their features (in the sets of $\mathbf{Fu_{xx}}$ and $\mathbf{Ru_{xx}}$) and the weight array (see fifth step in Fig. 3). In the sixth step, the knowledge and useful data of users are finally stored in their knowledge models for future use.

4. Applying the AEEC user modeling approach to the probabilistic and instance-based classification methods

4.1. A probabilistic method: development of the Bayes knowledge classifier

We developed a *Bayes knowledge classifier* in AEEC and introduced in the reference studies [5,8,9]. The classifier is also summarized in the following paragraphs briefly.

Naive Bayes Classifier (NBC) is a probabilistic and supervised learning method frequently used in many text classification problems, in user modeling tasks, and in medical diagnosis problems. An NBC assumes that the features affecting the class of a state are independent. Therefore, the NBC is also known as 'feature independent model'.

Let ⟨$a_1, a_2, a_3, \ldots, a_q$⟩ be an independent feature set 'A' and the class of hypothesis came out depending on this feature set be represented by 'C'. According to Bayes's theory [10,33], the probability model is:

$$P(C|a_1, a_2, a_3, \ldots, a_q) = \frac{P(C)P(a_1, a_2, a_3, \ldots, a_q|C)}{P(a_1, a_2, a_3, \ldots, a_q)} \quad (1)$$

$$P(a_1, a_2, a_3, \ldots, a_q = \arg\max_{C_{j\in C}} P(c_j) \prod_{a_i \in A} P(a_i|C_j) \quad (2)$$

The expression in Eq.2 can be rewriten to create Bayes knowledge classifier. $\mathbf{Fu_c}$ ⟨STG, SCG, PEG⟩ and $\mathbf{Ru_c}$ ⟨STR, LPR⟩ represent the independent feature set and ⟨UNS⟩ represents class of hypothesis for the Bayes knowledge classifier. The probabilities of students' knowledge class ⟨UNS⟩ can be calculated from:

$$P(UNS|STG, SCG, STR, LPR, PEG) = \frac{P(UNS)P(STG, SCG, STR, LPR, PEG|UNS)}{P(STG, SCG, STR, LPR, PEG)} \quad (3)$$

Table 6 illustrates the attributes and hypothesis and their labels affecting the knowledge class of students. Labeling the attributes is performed by rule based system and labeling the knowledge class is performed by the classifier algorithm. Please refer to the references [8–10] for detailed information about the *Bayes knowledge classifier*.

4.2. An instance-based method: development of k-NN knowledge classifier

In this subsection, domain dependent data of users and user modeling approach of AEEC are combined with *k*-NN classifier. The combination is also called *k*-NN knowledge classifier.

k-NN algorithm is a memory-based method frequently used for regression and classification tasks. Just like NBC, *k*-NN algorithm is

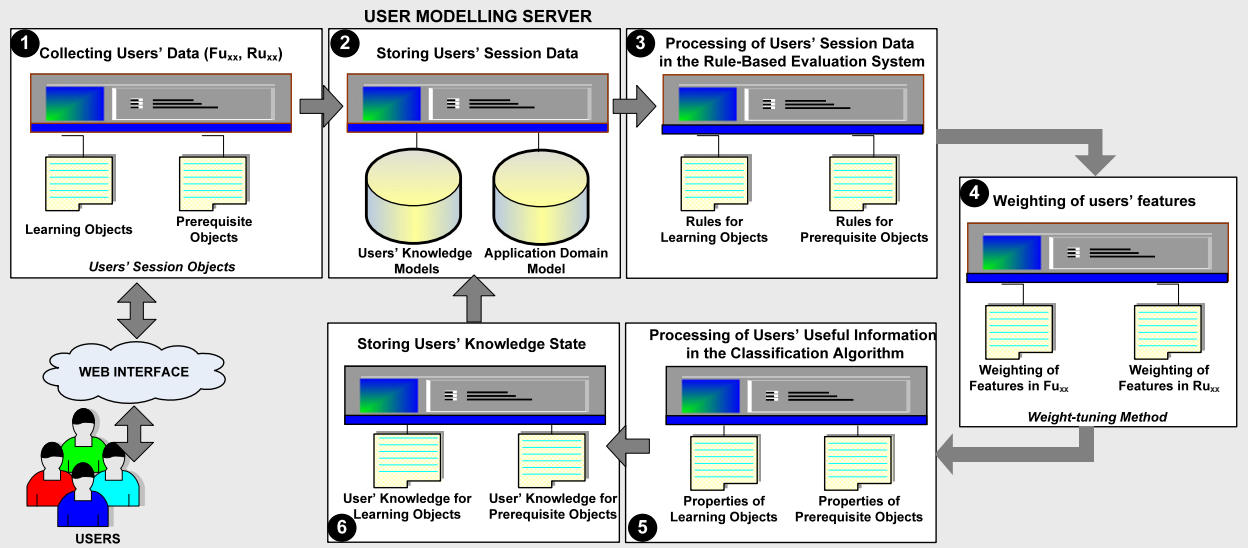

Fig. 3. The architecture of proposed IKC approach.

Table 6
Representation of the target class together with the features affecting the users' knowledge in NBC algorithm.

Attributes for NBC	The class of hypothesis (users' knowledge class)
Labels for ⟨STG, SCG, STR, LPR, PEG⟩	Classes for UNS
* very low, * low, * middle, * high	* beginner, * intermediate, * expert, * advanced

a supervised learning algorithm. The three key parameters of the classification process are the integer '*k*', training set and distance metric. The '*k*' value is determined by the developer in algorithm design process. It represents the number of nearest neighbors that will be searched within the training set. In order to classify a new instance sample, the distances among all the samples of the training set and the instance sample are measured. 'Euclidean' (EU), 'Manhattan' (MA) and 'Minkovski' (MI) are the most widely used distance metrics [10,33]. After that, the classes of *k*-number of neighbors are determined if they are the closest to the instance sample. For this purpose, the majority-vote method is commonly used to determine the class of the instance sample among the *k*-neighbors [10,31].

Let $\mathbf{x}$ and $\mathbf{y}$ be two points in an n-dimensional space and in Cartesian coordinate system $\mathbf{x} = (x_1, x_2, x_3, \ldots, x_n)$ with $\mathbf{y} = (y_1, y_2, y_3, \ldots, y_n)$. The distance $d(\mathbf{x}, \mathbf{y})$ measurement between $\mathbf{x}$ and $\mathbf{y}$ points is calculated from Eqs. (4)–(6). EU distance metric is a formula used for measuring the length between two points in Cartesian coordinate system. The length between $\mathbf{x}$ and $\mathbf{y}$ points can be converted into metric distance by means of 'EU' as shown in Eq. (4), 'MA' as shown in Eq. (5), 'MI' as shown in Eq. (6):

$$\textit{Eucledion Distance}; d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{4}$$

$$\textit{Manhattan Distance}; d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |\mathbf{x}_i - \mathbf{y}_i| \tag{5}$$

$$\textit{Minkowski Distance}; d(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} |\mathbf{x}_i - \mathbf{y}_i|^m \right)^{1/m} \tag{6}$$

Table 7 shows the domain dependent data or features of users and the labels of users' knowledge class of users in *k*-NN knowledge classifier. *k*-NN classifies users' knowledge (the second column in Table 7) for a learning object depending on the real-values of their domain dependent data (the first column in Table 7). Different from the NBC algorithm, quantitative values are used instead of categorical values for representing users' domain dependent data.

Fig. 4 shows the target class distribution (five items in the class distribution are 'beginner', seven of them are 'intermediate', eight of them are 'expert' and six of them are 'advanced') of the *k*-NN training set with 26 observations. In Fig. 4, K_{UNS} is a query or test observation representing the domain dependent data (STG_{Kuns}, SCG_{Kuns}, STR_{Kuns}, LPR_{Kuns}, PEG_{Kuns}) of a user about a learning object. The knowledge class of users is predicted in four steps. The steps are summarized as follows.

i. Determining the *k*-value (the number of nearest neighbors for K_{UNS} observation).
ii. Calculate the distances between the K_{UNS} observation and the training instances.

EU, MA and MI metrics are used to determine the nearest neighbors of K_{UNS} observation from among the training observations. According to EU metric given in Eq. (4) the measurement of distances in five-dimensional features space is calculated from the following equation:

$$\prod_{l=1}^{26} dK_{UNS[l]} = \sqrt{(STG_{Kuns} - STG_{[l]})^2 + (SCG_{Kuns} - SCG_{[l]})^2 + (STR_{Kuns} - STR_{[l]})^2 + (LPR_{Kuns} - LPR_{[l]})^2 + (PEG_{Kuns} - PeG_{[l]})^2} \tag{7}$$

Table 7
Representation of the features and the target class, affecting the user knowledge state, in *k*-NN knowledge classifier.

Features for *k*-NN (users' domain dependent data)	Target class (users' knowledge)
Values of ⟨STG, SCG, STR, LPR, PEG⟩ Normalized values between 0 and 1	Classes of UNS 'beginner', 'intermediate', 'expert', 'advanced'

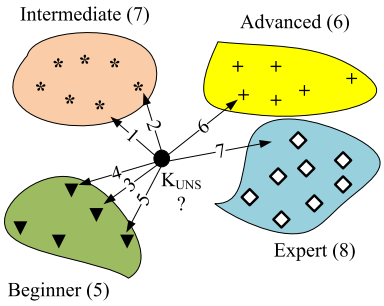

Fig. 4. The *k*-neighbors of K_{UNS} test observation in the training space with 26 samples.

By means of using other distance metrics given in Eqs. (5) and (6), the distances between the user' feature set $\langle STG_{Kuns}, SCG_{Kuns}, STR_{Kuns}, LPR_{Kuns}, PEG_{Kuns}\rangle$ and the other feature sets ($\langle STG_{[I]}, SCG_{[I]}, STR_{[I]}, LPR_{[I]}, PEG_{[I]}\rangle$) in the training space can similarly be calculated.

iii. Sorting distances $dK_{UNS[I]}$ and determining of nearest neighbors of K_{UNS} observation

In Fig. 5, the arrows show the nearest 7-neighbors for K_{UNS} in training space. The numbers (1–7) are used to show the closeness level of training observations. Some examples are given below:

If k = 3 then the class distribution: the class of two observations is 'intermediate' (arrows 1 and 2) and the class of one observation is 'beginner' (arrow 3).

If k = 7 then the distribution of the class is: the class of two observations is 'intermediate' (arrows 1 and 2), the class of three observations is 'beginner' (arrows 3, 4 and 5), the class of one observation is 'expert" (arrow 7) and the class of seventh observation is 'advanced' (arrow 6).

iv. Make a majority vote among the nearest neighbors in order to determine the class of K_{UNS} observation.

According to majority vote method, the knowledge class (UNS) of user is determined as 'intermediate' (with two votes) for $k = 3$ and the class of the UNS is determined as 'beginner' (with three votes) for $k = 7$

4.3. Development of intuitive knowledge classifier

In this subsection, the instance-based *k*-NN knowledge classifier is combined with the genetic algorithm-based (GA) weight-tuning method. Firstly, weight-tuning method including genetic algorithm-based (GA) intuitive searching approach is introduced. The weight-tuning method is used to explore the real-valued weights of students' domain dependent data/five features of students in the sets of $\mathbf{Fu_c}$ ⟨STG, SCG, PEG⟩ and $\mathbf{Ru_c}$ ⟨STR, LPR⟩ on their knowledge class ⟨UNS⟩. The best weight-values $\langle W_{STG}, W_{SCG}, W_{PEG}, W_{STR}, W_{LPR}\rangle$ are stored and used to measure similarities among the students' data by the IKC algorithm. Fig. 5 depicts the processing flow of the IKC algorithm. The IKC includes the following steps:

(i) it learns the data structure and features of instructional objects from application domain model.
(ii) according to the data structure and features of objects, it obtains a training set including samples of domain dependent data of students from their knowledge model.
(iii) the weight-tuning method begins by assigning real-valued weights in the interval [0–1] to features of students. Weight arrays are constructed and updated to explore the best weight values by the method in this process. After that, IKC considers the training set and weight arrays in the classifying the knowledge of students. Once the weights provide a significantly classification performance over the validation set then the best weights are stored and weight searching process is terminated. Finally, the stored weights are used to classify the new samples of students' data.

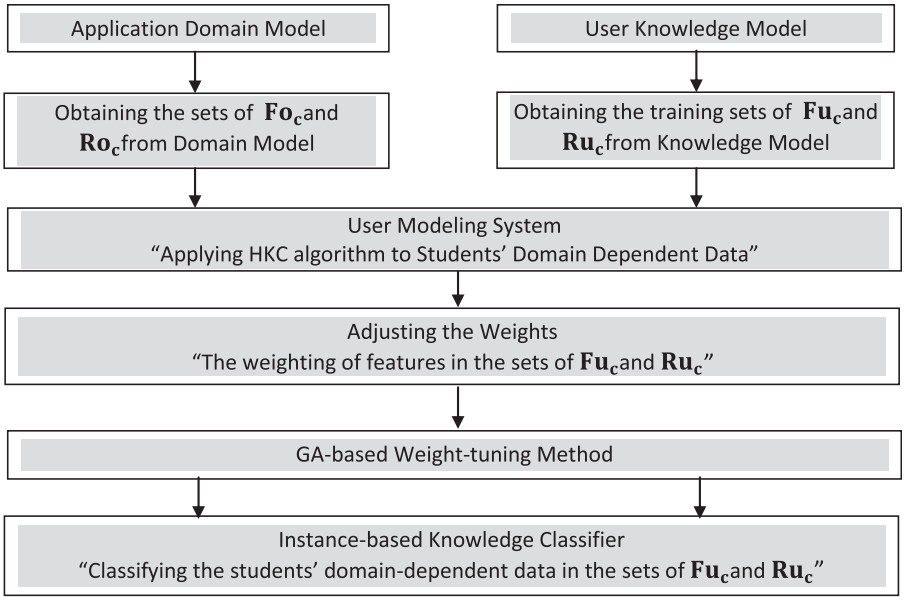

Fig. 5. The weighting of students' features in the proposed IKC-based approach.

Table 8
The genes of chromosome in GA-based weight-tuning method.

Gene 1	The weight for STG (W_{STG})
Gene 2	The weight for SCG (W_{SCG})
Gene 3	The weight for STR (W_{STR})
Gene 4	The weight for LPR (W_{LPR})
Gene 5	The weight for PEG (W_{PEG})
Gene 6	The fitness value of individual (FV)

4.3.1. Genetic algorithm-based weight-tuning method

Two basic steps of genetic algorithm (GA)-based weight-tuning method coding of individuals in population and updating population are given in following sentences.

(i) coding of individuals: The students' features affecting their knowledge are given in Table 7. The weight of each feature can be represented in the form of $\langle W_{STG}, W_{SCG}, W_{PEG}, W_{STR}, W_{LPR}\rangle$ with a weight vector depending on the IKC approach. An individual (chromosome) is coded as in Table 8. The genes of an individual correspond to the weights of domain dependent data of students and the fitness value of it. The fitness of an individual is represented by a fitness value (FV).

If it is rewritten as in Eq. (7), Eq. (8) is obtained by considering weighted features as:

$$\prod_{l=1}^{26} dw_{K_{uns[l]}} = \sqrt{W_{STG} * (STG_{Kuns} - STG_l)^2 + \ldots W_{PEG}(PEG_{Kuns} - PEG_l)^2} \tag{8}$$

According to Eq. (8), when the weighted features are included in the similarity measurements and then the distances among the students change. As distances change, the classification accuracy and the performance of IKC algorithm over the validation set also change. The fitness values of individuals show the performance of IKC corresponding to their weight values.

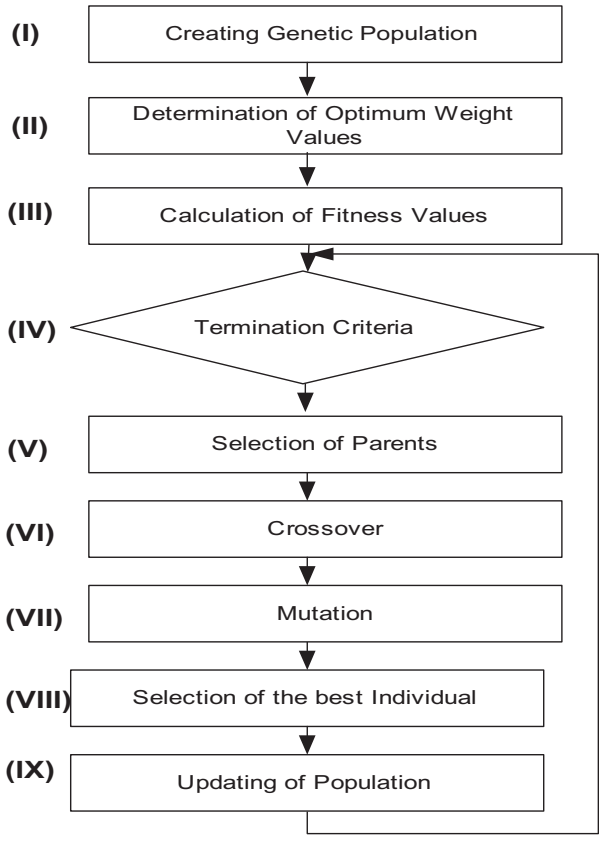

Fig. 6. The steps for creating and updating of population.

(ii) creating and updating of population: The flow diagram that shows the creating and updating steps of a population is given in Fig. 6.

- creating population: h-item individual is created to constitute the population. The individuals have the genetic codes that are shown in Table 8. The matrix form of population ($P_{IKC}[h,6]$) constituting h-item individual is given as follows:

$$P_{IKC\,[h,6]} \equiv \begin{bmatrix} W_{STG\,[0,0]} & W_{SCG\,[0,1]} & W_{STR\,[0,2]} & W_{LPR\,[0,3]} & W_{PEG\,[0,4]} & FV_{[0,5]} \\ & & \vdots & & & \\ W_{STG\,[h,0]} & W_{SCG\,[h,1]} & W_{STR\,[h,2]} & W_{LPR\,[h,3]} & W_{PEG\,[h,4]} & FV_{[h,5]} \end{bmatrix} \tag{9}$$

Weights are coefficients having numeric value in the interval [0,1]. The type of data is double. *Fitness value* (*FV*) is a real value representing the fitness of chromosome for problem. It might have a value in the interval [0,1].

- determination of the best weight values: Each observation object or domain dependent data of students has consisted of feature vector with six dimensional according to Tables 7 and 8. The feature vector is represented with tuples $\langle STG, SCG, STR, LPR, PEG\rangle$ and $\langle UNS\rangle$. It supposes that **U** is a training set of sample observations stored in students' database and ***l*** is the sample number in this set. The training set **U** is given as follows:

$$\mathbf{U}_{[l,6]} \equiv \begin{bmatrix} STG_{[0,0]} & SCG_{[0,1]} & STR_{[0,2]} & LPR_{[0,3]} & PEG_{[0,4]} & UNS_{[0,5]} \\ & & \vdots & & & \\ STG_{[l,0]} & SCG_{[l,1]} & STR_{[l,2]} & LPR_{[l,3]} & PEG_{[l,4]} & UNS_{[l,5]} \end{bmatrix} \tag{10}$$

The pseudocode of IKC algorithm to explore the optimum weight values is given in Algorithm 1.

Algorithm 1. The pseudocode of GA-based weight-tuning method to explore the optimum weight values of the domain dependent data/features of users

1: ***for tt = 0 to h*** (calculate the fitness values of the individuals in $\mathbf{P_{IKC[tt]}}$)
2: ***for jj = 0 to l*** (use the all observations in the set of **U**)
3: Use the Eq. (8) for EU metric and calculate the distances between the jjth observation $\mathbf{U_{jj}}$ and the other observations in set **U** depending on the real-values of observations and weight values of $\mathbf{P_{IKC[tt]}}$
4: Determine the class of jjth observation U_{jj} using the k-NN knowledge classifier
5: Compare the determined class of U_{jj} with real class of it
   a. If the comparison is true then the fitness value of ***tt***th individual in the $\mathbf{P_{IKC[tt]}}$ is increased
   b. Else the fitness value of ***tt***th individual in the $\mathbf{P_{IKC[tt]}}$ is decreased
6: for qq = 0 to generation number of population (termination criteria)
7: Select the parents
8: Achieve the crossover and the mutation operations
9: Calculate the fitness values of new individuals
10: Select most valuable individual
11: Update $\mathbf{P_{IKC}}$ population
12: Finalize the searching process and save the weight values of the most suitable individual in the set of $\mathbf{P_{IKC[tt]}}$

Table 9
Rule-based conversion table (the rules are used to convert the categorical data to numerical values) [5].

Feature	Rule
STG SCG	$\begin{cases} 0 < \text{STG or SCG} < 0.25 \leftarrow \text{“very low”} \\ 0.25 \Leftarrow \text{STG or SCG} < 0.33 \leftarrow \text{“low”} \\ 0.33 \Leftarrow \text{STG or SCG} < 0.5 \leftarrow \text{“middle”} \\ 0.5 \Leftarrow \text{STG or SCG} \leftarrow \text{“high”} \end{cases}$
STR	$\begin{cases} 0 < \text{STR} < 0.3 \leftarrow \text{“very low”} \\ \text{STR} \Rightarrow 0.3 \wedge \text{STR} < 0.5 \leftarrow \text{“low”} \\ \text{STR} \Rightarrow 0.5 \wedge \text{STR} < 0.7 \leftarrow \text{“middle”} \\ \text{STR} \Rightarrow 0.7 \leftarrow \text{“high”} \end{cases}$
LPR PEG	$\begin{cases} 0 < \text{LPR or PEG} < 0.25 \leftarrow \text{“very low”} \\ \text{LPR or PEG} \Rightarrow 0.25 \wedge \text{LPR} < 0.33 \leftarrow \text{“low”} \\ \text{LPR or PEG} \Rightarrow 0.33 \wedge \text{LPR or PEG} < 0.66 \leftarrow \text{“middle”} \\ \text{LPR or PEG} \Rightarrow 0.66 \leftarrow \text{“high”} \end{cases}$

- calculation of fitness value: It assumes that; ‘e’ denotes the number of misclassified observations corresponding to any individual in the set of $\mathbf{P_{IKC[tt]}}$. Fitness value of this individual is calculated as $FV = 1/\mathbf{e_c}$. Fitness values of all individuals in population are calculated similarly.
- termination criteria: In this process, the best individual providing the genetic reproduction finalization criteria among individuals in $\mathbf{P_{IKC[tt]}}$ population is researched. Termination criteria might be a definite fitness value or a generation number. After the termination of reproduction, the weights of individual having the best fitness value in $\mathbf{P_{IKC[tt]}}$ are chosen as the best weight values of population.
- the selection of parents: each individual in population represents a solution for problem. Reproduction operator is used to create new individuals in population. The methods of ‘Tournament’ and ‘Roulette Wheel’ are implemented to select the parents in this study.

Table 10
Training data set samples for *k*-nn knowledge classifier and IKC algorithm.

No	STG	SCG	STR	LPR	PEG	UNS
1	0.080	0.325	0.620	0.940	0.560	Advanced
2	0.265	0.600	0.280	0.660	0.070	Beginner
3	0.640	0.550	0.150	0.180	0.630	Expert

Table 11
Training data set sample for Bayes knowledge classifier.

No	STG	SCG	STR	LPR	PEG	UNS
1	Very low	Low	Middle	High	Middle	High
2	Low	High	Very low	High	Very low	Very low
3	High	High	Very low	Very low	Middle	Middle

Table 12
The parameters of IKC algorithm.

Individual number in population	Population size	Parent selection method	Crossover method
150	12,000	Roulette wheel	Flip bit, boundary non-uniform, uniform
Mutation coefficient	Mutation method	Neighbor value (k)	Distance metric
interval [0.01;0.001]	Single point, two points, inversion	1, 3, 5,7	EU, MI, MA

- crossover: It is process that the execution of permutation process vice versa among the genes of children. Thanks to this, the genetic codes of individuals have been changed. In this article, the three crossing methods (Flip bit, Boundary and Uniform) are implemented.
- mutation: The genetic codes of child individuals are changed. It is used to maintain genetic diversity in $\mathbf{P_{IKC[tt]}}$ population. In this study, the random-mixed of mutation methods named ‘bit inversion’, ‘swaps’ and ‘single point mutation’ are used. As reached the individual target in population the reproduction among the individuals is continued.
- the selection of the best individual: It is that individual the best fitness value through children is selected.
- updating of population: It is that individual the lowest fitness value in population is replaced by the best individual determined in Fig. 6.

4.3.2. Classification in IKC

After weighting the features, the knowledge of students for a learning object is classified by IKC depending on their domain dependent data $\langle STG, SCG, STR, LPR, PEG\rangle$ about that object and the weight values of features $\langle W_{STG}, W_{SCG}, W_{STR}, W_{LPR}, W_{PEG}\rangle$. The class of users is predicted in five steps. The steps are summarized in the Algorithm 2.

Algorithm 2. The steps of knowledge classification in IKC algorithm

Training set ($U_{[I]}$): $[STG_{[l]}, SCG_{[l]}, STR_{[l]}, LPR_{[l]}, PEG_{[l]}]$
I: the number of observations in the training set of $U_{[I]}$
The domain dependent data of users for a learning object: $[STG_{Kuns}, SCG_{Kuns}, STR_{Kuns}, LPR_{Kuns}, PEG_{Kuns}]$
Weights of user’ features (weight array): $[W_{STG}, W_{SCG}, W_{STR}, W_{LPR}, W_{PEG}]$
$dw_{KUNS[n]}$: the distance between the user and nth sample observation in the set of $U_{[I]}$
1: *determining the number of nearest neighbors (k-value)*
2: *selection of similarity metric*
3: *similarity measurements*
for ii = 0 to l
• *use the distance method* (Eq. (8) *for EU metric*): $f_{distance}$ *(user’ data,* $U_{[l]}$*, weight array) and calculate the distances between the user and the observations in the set of* **U**.
• *create a distance array in the form of* ($dw_{KUNS[l]} = [d_1, d_2, d_3, \ldots, d_I]$)
4: *sort the distances in* $dw_{KUNS[l]}$ *and find the nearest neighbors*
5: *classify the user’ knowledge: make a majority vote among the classes of the k-nearest neighbors in order to determine the user’ knowledge class for the learning object*

5. Experimental study

In this section, the classification results of the well-known probabilistic and instance-based user modeling algorithms are compared with the proposed IKC method. The classification accuracy of the algorithms over the validation or test data set was compared for different distance metrics, weight values and *k*-values. For this purpose, a data set consisting of 1024 samples were used to train and validate the algorithms. While categorical data was used in the probabilistic Bayes knowledge classifier algorithm (please see Table 6); the quantitative/numerical values were used in the instance-based *k*-NN knowledge classifier and the IKC algorithm (please see Table 7). The same training and validation data sets were used in the experimental studies. Instance-based algorithms use numeric values in training process. Therefore, a rule-based

Table 13
The best weight values for different k-values and distance metrics.

Metric	*k*-Value	Gene 1 W_{STG}	Gene 2 W_{SCG}	Gene 3 W_{STR}	Gene 4 W_{LPR}	Gene 5 W_{PEG}	Gene 6 FV
EU	1	0.54203	0.01075	0.02407	0.05033	0.95930	0.25
EU	3	0.05547	0.01184	0.01699	0.03167	0.95214	0.33
EU	5	0.04599	0.01597	0.03045	0.05562	0.82836	0.33
EU	7	0.04001	0.02222	0.02205	0.12386	0.77102	0.50
MA	1	0.61299	0.05427	0.05347	0.23129	0.86429	0.20
MA	3	0.07080	0.08460	0.06206	0.14096	0.76249	0.33
MA	5	0.14503	0.05685	0.21896	0.21755	0.95365	0.50
MA	7	0.09128	0.10922	0.18249	0.37614	0.78345	0.50
MI	1	0.00008	0.00038	0.98214	0.00040	0.82122	0.20
MI	3	0.64886	0.00005	0.00005	0.00279	0.57515	0.33
MI	5	0.00429	0.00043	0.00043	0.89958	0.98539	0.17
MI	7	0.05323	0.00102	0.00103	0.52409	0.88878	0.17
Average		0.19251	0.03063	0.13285	0.22119	0.84544	0.32

conversion table has been adopted depending on the expert opinions (see Table 9). In Table 9, the rule column represents the numerical intervals the quantitative equivalent of categorical data ('very low', 'low', 'middle', and 'high'). For instance, if the category of STG is *very low*, then its numerical equivalent is a real value in the interval [0,0.25]. The exact value of a feature is determined in two steps as given below:

i. The feature and its category are determined. The feature set: ⟨STG, SCG, STR, LPR, PEG⟩ and categories of features: 'very low', 'low', 'middle', and 'high'.
ii. The exact value of the feature is determined by using the information in conversion table. Table 9 shows the minimum and maximum values which are corresponding to the category of the feature.

5.1. The data sets

A random subset with 258 observations of the available data set was selected to generate the training set of algorithms. Table 10 shows three samples of training observations of the *k*-NN knowledge classifier and IKC algorithm.

After processing the training data set (in Table 10) composed of numerical data through rule-based conversion system, the training set of Bayes knowledge classifier is obtained as Table 11. A random subset with 145 observations of the available data set was selected to generate the validation set of algorithms.

5.2. The parameters of GA-based weight-tuning method

The parameters of genetic algorithm and *k*-NN knowledge classifier and their typical settings are illustrated in Table 12. The parameters determine the general conditions of GA-based weight tuning method.

5.3. The best weight values of users' features

Weight-tuning method has searched optimum weight values of domain dependent data (features) of users. Total number of the experiments or the created populations is 12. The experiments

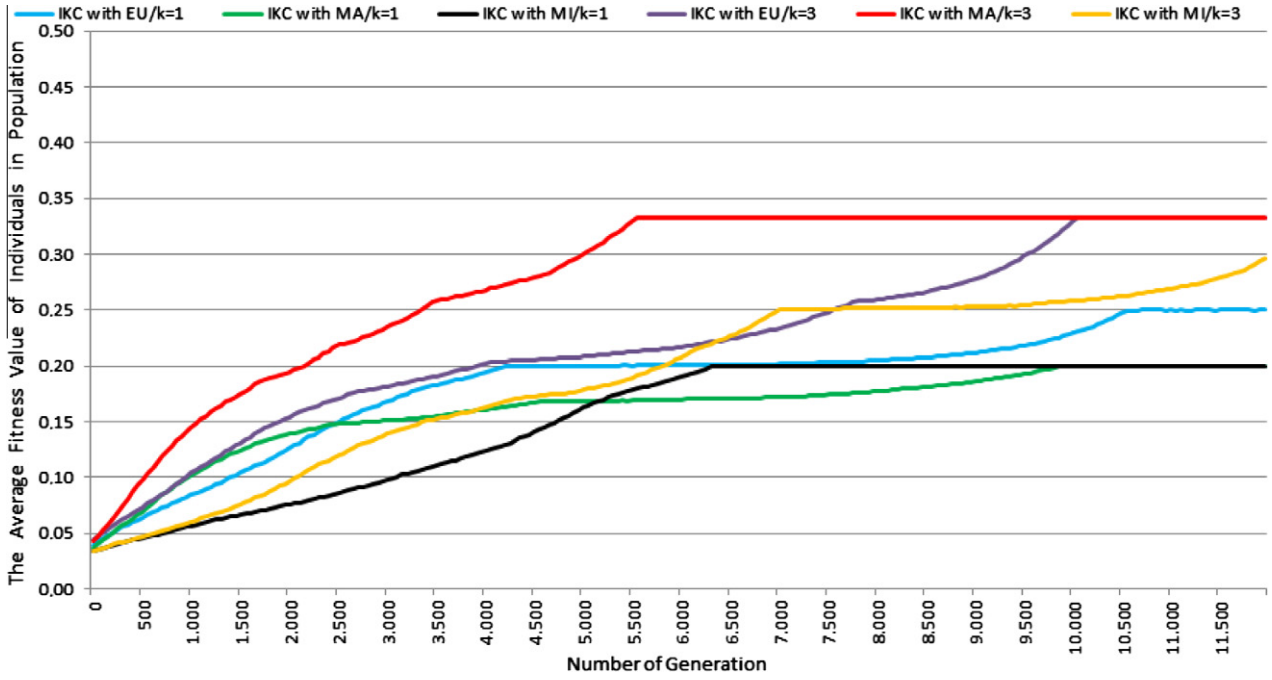

Fig. 7. Change of fitness value of populations depending on different metrics and *k*-values (1 and 3).

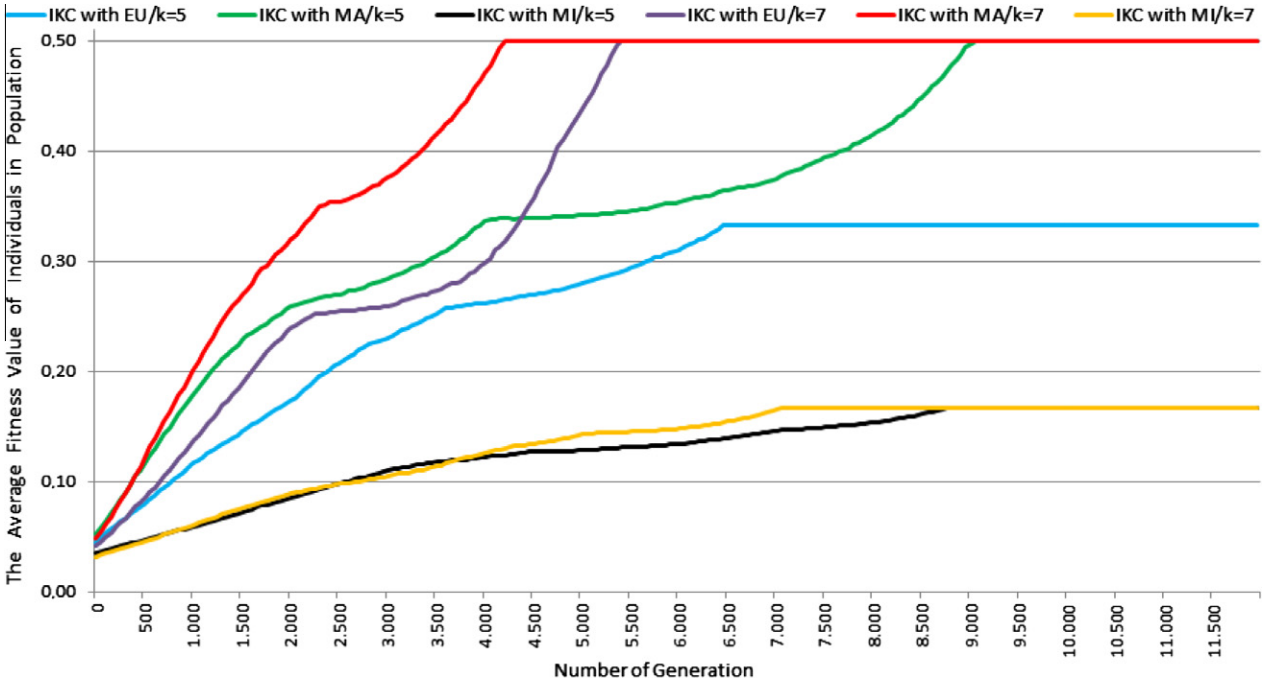

Fig. 8. Change of fitness value of populations depending on different metrics and *k*-values (5 and 7).

have been performed by the IKC algorithm for four different *k*-values (1, 3, 5, 7) and three metrics (EU, MA, MI). The genes of the best individuals each of population or experimental study are listed in Table 13. The genes present the weight values and fitness value of the best individual for a specific *k*-value and distance metric.

When the weight values in Table 13 are analyzed, it is seen that the W_{LPR} and W_{PEG} have larger weight values than the others. It shows that the features of students (the LPR and PEG) are considerably more efficient than the other three features (STG, SCG, STR) on the knowledge class of users. The curves of test set were presented in Figs. 7 and 8. The figures depict the change of fitness value of populations for three different distance metrics and four *k*-values (1, 3, 5, and 7). The graphs show how generation number, distance metric and *k*-value influence the fitness value of IKC.

5.4. Comparing knowledge classifiers

One experimental comparison of the knowledge classifiers was provided. After training on 258 observations/samples, the average number of misclassified observations/samples, the percentage of average error rates and the average classification accuracy of three knowledge classifiers were measured for 145 test observations (validation set), four *k*-values (1, 3, 5, 7) and three distance metrics (EU, MA, MI) (see Table 14).

Table 14
Comparing the performances of algorithms.

Algorithm	Bayes knowledge classifier	*k*-NN knowledge classifier			Proposed IKC		
		EU	MA	MI	EU	MA	MI
The average number of misclassified observations	38	26.2	21.7	30.5	3	3	5
The percentages of average error rates (%)	26.2	18.1	15	21	2.1	2.1	3.5
The classification accuracy (%)	73.8	81.9	85	79	97.9	97.9	96.5

The proposed IKC method achieved an average classification accuracy of 97.5% over the validation set, whereas the classification accuracy of the other approaches ranged from 73.8% to 85%. Consequently, IKC approach has produced a considerable improvement: the rate of improvement is 23.7% for the Bayes knowledge classifier and 15.5% for the instance-based *k*-NN knowledge classifier.

The percentages of average error rates are displayed in Fig. 9. It can be seen that the error rate of knowledge classifier over the validation set is significantly reduced in IKC algorithm. In other words, the classification accuracy of IKC is significantly higher than the other knowledge classification algorithms for all *k*-values and distance metrics.

6. Conclusions

This paper presents a powerful, efficient and simple 'Intuitive Knowledge Classifier' method successfully and proposes to model the domain dependent data of users. A domain independent object model, the user modeling approach and the weight-tuning method were combined with instance-based classification algorithm to improve classification performances of well-known the Bayes and the *k*-nearest neighbor-based methods.

The proposed method consists of integrations of the domain-independent object model, the robust and the effective user modeling approach of AEEC, the instance-based classification algorithm and a novel weight-tuning method. Firstly, the generic object model eliminates the application dependency of user modeling system. Secondly, the user modeling approach of AEEC was used to determine the type and attributes of domain dependent data stored and evaluated by the user modeling system. Thirdly, the combination of the well-known instance-based classification algorithm and the user modeling system of AEEC has provided strong foundations, domain independence, simplicity and robust architecture for UMSs. Thus, the instance-based knowledge classifier was developed successfully.

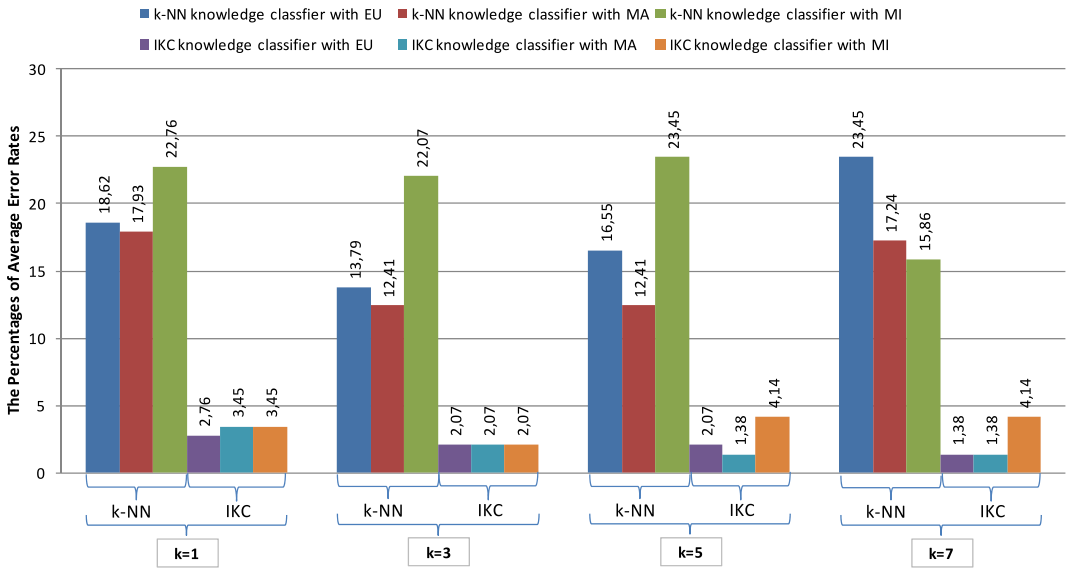

Fig. 9. The percentages of average error rates of instance-based methods for four *k*-values and three metrics.

The proposed knowledge classifier intuitively explores the optimum weight values of students' features on their knowledge class first. It then measures the distances among the students depending on their data and the values of weights. Finally, it uses the dissimilarities in the classification process to determine their knowledge class. The weighting of domain dependent data, each of features of students or each of features also provided a mechanism for suppressing the impact of irrelevant and less relevant factors in the similarity measurements. In other words, each of features has an effect differently when calculating the distances among the users. The weighted features of users (weighted feature set) might be used for classification purposes in data analysis.

The experimental studies have shown that the weighting of domain dependent data of students and combination of user modeling algorithms and population-based searching approach play an essential role in classifying performance of user modeling system. The proposed system improves the classification accuracy of instance-based user modeling approach for all distance metrics and different *k*-values. Intuitive weighting method improves the classification accuracy of instance-based knowledge classifier for all distance metrics and different *k*-values. Moreover, the results show that a significant improvement is achieved by using IKC method instead of the classic instance-based methods or Bayes-based probabilistic method. The proposed method has achieved an average classification accuracy of 97.5%, whereas the performance of the other approaches ranged from 73.9% to 82%. Consequently, IKC approach has improved the error rate 23.6% comparing to the Bayes knowledge classifier, and 15.5% comparing to the instance-based *k*-NN knowledge classifier. The proposed user modeling approach provides a useful conceptual framework to explore the effectiveness/weight values of domain dependent data of students on their knowledge class. The IKC algorithm has similarities with *k*-NN knowledge classifier in points of the classification simplicity, the parametric structure and the rapid response time. However, the use of IKC requires additional tasks, such as calculation of weight values and also more effort for programming. The weighting process is executed in off-line process by the proposed IKC. Thus it has not a negative effect in the online classification process. Consequently, the basic design issues and approaches, the development process, the implementation and the evaluation of a powerful and effective user modeling system were throughout presented in this article.

In future studies, different distance metrics and weight-tuning methods based on fuzzy logic-based distance metric and bee-colony based modern heuristic methods might be combined with the proposed IKC-based method to improve the approaches presented in this work.

References

[1] V. Tsiriga, M. Virvou, A Framework for the Initialization of Student Models in Web-based Intelligent Tutoring Systems, User Modeling and User-Adapted Interaction, vol. 14, Kluwer Academic Publishers, 2004. pp. 289–316.
[2] J. Zeng, S. Zhang, C. Wu, A framework for WWW user activity analysis based on user interest, Knowledge-Based Systems 21 (2008) 905–910.
[3] J. Armani, L. Botturi, Bridging the gap with MAID: a method for adaptive instructional design, in: Advances in Web-Based Education: Personalized Learning Environments, Information Science Publishing, Hershey, PA, USA, 2005, pp. 147–178.
[4] A. Cristea, A. De Mooij, LAOS: layered WWW AHS authoring model and its corresponding algebraic operators, in: Proceedings of WWW'03, Alternate Education track, Budapest, Hungary, 2003, pp. 1-10.
[5] H.T. Kahraman, Designing and application of web-based adaptive intelligent education system, Ph. D. Thesis, Institute of Science and Technology, Ankara, 2009.
[6] G. Castillo, J. Gama, A.M. Breda, An adaptive predictive model for student modelling, in: Advances in Web-Based Education: Personalized Learning Environments, Information Science Publishing, Hershey, PA, USA, 2005, pp. 70–93. Chapter IV.
[7] B. Qin, Y. Xia, S. Wang, X. Du, A novel Bayesian classification for uncertain data, Knowledge-Based Systems 24 (2011) 1151–1158.
[8] I. Colak, S. Sagiroglu, H.T. Kahraman, A user modeling approach to web based adaptive educational hypermedia systems, in: Seventh International Conference on Machine Learning and Applications, IEEE Computer Society, 2008, pp. 694–699.
[9] H.T. Kahraman, I. Colak, S. Sagiroglu, A web based adaptive educational system, in: The Sixth International Conference on Machine Learning and Applications (ICMLA'07), IEEE Computer Society, 13–15 December 2007, pp. 1–6.
[10] T.M. Mitchell, Machine Learning, McGraw-Hill Science/Engineering/Math, 1997. pp. 154–184.
[11] M.C. Desmarais, P. Meshkinfam, M. Gagnon, Learned Student Models with İtem to İtem Knowledge Structures, User Modeling and User-Adapted Interaction, vol. 16, Springer Science + Business Media B.V., 2006. pp. 403–434.
[12] M.L. Kherfi, Review of human-computer interaction issues in image retrieval, in: Shane Pinder (Ed.), Advances in Human-Computer Interaction, I-Tech Education and Publishing KG, Vienna, Austria, 2008, pp. 215–240.
[13] G.I. Webb, M.J. Pazzani, D. Billsus, Machine Learning for User Modeling, User Modeling and User-Adapted Interaction, vol. 11, Kluwer Academic Publishers, 2001. pp. 19–29.
[14] J.D. Kelly, L. Davis, A hybrid genetic algorithm for classification, IJCAI-91 (1991) 645–650.
[15] F.G. Halasz, M. Schwartz, The Dexter Hypertext Reference Model, NIST Hypertext Standardization Workshop Gaithersburg, 1990, pp. 1–40.

[16] P. DeBra, G.J. Houben, H. Wu, AHAM: a Dexter-based reference model for adaptive hypermedia applications, in: Proceedings of the 10th ACM Conference on Hypertext and Hypermedia, ACM Press, New York, 1999, pp. 147–156.
[17] H.T. Kahraman, S. Sagiroglu, I. Colak, A novel model for web-based adaptive educational hypermedia systems: SAHM (Supervised Adaptive Hypermedia Model), in: Computer Applications in Engineering Education, John Wiley & Sons, 2010 (Published online in Wiley InterScience, doi:http://dx.doi.org/10.1002/cae.20451).
[18] S. Sagiroglu, I. Colak, H.T. Kahraman, Transition to adaptive educational hypermedia systems from web-based educational systems: review of design approaches for the AEHS, Journal of the Faculty of Engineering and Architecture of Gazi University 23 (4) (2008) 837–852.
[19] P. Brusilovsky, Developing adaptive educational hypermedia systems: from design models to authoring tools, in: Authoring Tools for Advanced Technology Learning Environment, Kluwer Academic Publishers, Dordrecht, 2003, pp. 377–409.
[20] K.A. Papanikolaou, M. Grigoriadou, G.D. Magoulas, H. Kornilakis, Towards new forms of knowledge communication: the adaptive dimension of a web-based learning environment, Computers and Education 39 (4) (2002) 333–360.
[21] K.A. Papanikolaou, M. Grigoriadou, H. Kornilakis, G.D. Magoulas, Personalizing the interaction in a web-based educational hypermedia system: the case of INSPIRE, User Modeling and User-Adapted Interaction, vol. 13, Kluwer Academic Publishers, 2003. pp. 213–267.
[22] Z.A. Pardos, N.T. Heffernan, Modeling individualization in a Bayesian networks implementation of knowledge tracing, in: 18th International Conference on User Modeling, Adaptation, and Personalization, UMAP 2010, Springer-Verlag, Berlin Heidelberg, 2010, pp. 255–266.
[23] P. Brusilovsky, E. Millan, User models for adaptive hypermedia and adaptive educational systems, The Adaptive Web in the Adaptive Web, vol. 4321, Springer-Verlag, Berlin Heidelberg, 2007. pp. 3–53.
[24] M.P. O'Mahony, B. Smyth, A classification-based review recommender, Knowledge-Based Systems 23 (2010) 323–329.
[25] R. Virgilio, R. Torlone, G.J. Houben, Rule-Based Adaptation of Web Information Systems, World Wide Web, vol. 10, Springer Science + Business Media, 2007. pp. 443–470.
[26] A.C. Martins, L. Faria, C. Vaz de Carvalho, E. Carrapatoso, User modeling in adaptive hypermedia educational systems, Educational Technology & Society 11 (1) (2008) 194–207.
[27] M. Bezold, Describing user interactions in adaptive interactive systems, in: 17th International Conference on User Modeling, Adaptation, and Personalization, UMAP 2009, Springer-Verlag, Berlin Heidelberg, 2009, pp. 150–161.
[28] C. Abela, C. Staff, S. Handschuh, Task-based user modelling for knowledge work support, in: 18th International Conference on User Modeling, Adaptation, and Personalization, UMAP 2010, Springer-Verlag, Berlin Heidelberg, 2010, pp. 419–422.
[29] J. Vassileva, A Task-Centered Approach for User Modeling in a Hypermedia Office Documentation System, Adaptive Hypertext and Hypermedia, Special Issue of User Modeling and User Adapted Interaction, vol. 6(2–3), Springer, Berlin, 1996, pp. 185–223.
[30] M. Simko, M. Bielikova, User modeling based on emergent domain semantics, in: 18th International Conference on User Modeling, Adaptation, and Personalization, UMAP 2010, Springer-Verlag, Berlin Heidelberg, 2010, pp. 411–414.
[31] M. Laguia, J.L. Castro, Local distance-based classification, Knowledge-Based Systems 21 (2008) 692–703.
[32] P. Cunningham, A taxonomy of similarity mechanisms for case-based reasoning, IEEE Transactions on Knowledge and Data Engineering 21 (11) (2009) 1532–1543.
[33] K. Ramamohanarao, H. Fan, Patterns Based Classifiers, World Wide Web, vol. 10, Springer Science + Business Media, 2007. pp. 71–83.