User Modeling and User-Adapted Interaction
https://doi.org/10.1007/s11257-024-09396-z

Personalized recommendations for learning activities in online environments: a modular rule-based approach

Radek Pelánek[1] · **Tomáš Effenberger**[2] · **Petr Jarušek**[2]

Received: 24 August 2023 / Accepted in revised form: 16 February 2024

Abstract
Personalization in online learning environments has been extensively studied at various levels, ranging from adaptive hints during task-solving to recommending whole courses. In this study, we focus on recommending learning activities (sequences of homogeneous tasks). We argue that this is an important yet insufficiently explored area, particularly when considering the requirements of large-scale online learning environments used in practice. To address this gap, we propose a modular rule-based framework for recommendations and thoroughly explain the rationale behind the proposal. We also discuss a specific application of the framework.

Keyword Recommender system · Education · Learning environment · Adaptive practice · Domain modeling

1 Introduction

Computerized learning environments have scaled from research prototypes to large-scale applications, particularly after the sudden boost of online learning during the global pandemic. Current online learning environments often offer students extensive content, and their offerings can be overwhelming. In such situations, personalized recommendations are helpful.

Personalization of learning can take many different forms. The content of learning environments can be considered at various levels of granularity, from steps within tasks

✉ Radek Pelánek
pelanek@fi.muni.cz

Tomáš Effenberger
tomas.effenberger@gmail.com

Petr Jarušek
jarusek.petr@gmail.com

1 Faculty of Informatics, Masaryk University Brno, Brno, Czech Republic

2 Umíme to, Brno, Czech Republic

Published online: 06 April 2024

that take a few seconds to whole courses that take weeks to finish. We focus on the level of learning activities—homogenous sequences of tasks that take a few minutes to finish, e.g., practicing the addition of decimal numbers or vegetable vocabulary.

There exists extensive research in several closely connected areas like recommending learning objects, student modeling, and sequential recommenders. For the specific area of activity recommendations, there is little directly relevant work that can serve as a guidance for designers of practical large-scale learning environments, who need to consider many aspects in the design of recommendations:

- support for various types of knowledge components, such as procedural rules and factual knowledge (Koedinger et al. 2012),
- support for various forms of exercises, such as selected response, constructed response, or interactive problem solving (Pelánek 2020),
- pedagogical strategies like spaced repetition and interleaved practice (Carpenter 2014; Dunlosky et al. 2013),
- prerequisites between topics (Chen et al. 2016),
- mapping of the content to grades or specific curricula,
- aspects connected to the practical usage of a learning environment in classrooms, such as homework,
- motivational factors, such as optimizing difficulty for reaching flow state (Nakamura and Csikszentmihalyi 2014; Lomas et al. 2013), personalization with respect to student interests, and others (Malone 1987).

In the research literature, these aspects are often addressed separately—researchers propose techniques that focus on optimizing specific aspects of learning environments. Such results are, however, often difficult to apply in conjunction with other aspects.

Our approach is based on the perspective of avoiding stupidity (Pelánek and Effenberger 2022). Rather than trying to optimize a single aspect of recommendations (e.g., optimal intervals for spaced repetition), we aim to ensure that all important aspects are taken into account, at least in their basic form. To this end, we propose a rule-based framework for recommending learning activities in computer-based learning environments.

The proposed recommendation framework encapsulates the formulation of objectives of the recommendations, modular data processing pipeline, principal realization of individual modules, and modular evaluation approach for continuous improvement. The proposed framework is based on IF-THEN rules, which are easily interpretable. Moreover, all intermediate steps in the modular pipeline have a clear interpretable meaning, and final recommendations can be naturally explained since each recommendation is a consequence of a specific interpretable rule. In the presentation of the framework, we clearly describe our rationale for various design choices, which combine insights from research endeavors and practical experience with running a large-scale learning environment.

The proposed framework is grounded in sound methodology and, at the same time, directly applicable in real-life settings. We have successfully used the framework to implement recommendations in a large-scale learning environment and provide a report on this case study. Moreover, the proposed framework is flexible enough to be applied in different learning environments. While previous research has intro-

Table 1 Learning units and personalization

Time scale	Learning unit	Personalization
10 s	Step	Adaptive hints
1 min	Task	Difficulty adjustment, personalized feedback
10 min	Activity	Mastery learning, activity recommendation
Hours	Course	Course recommendation

duced numerous educational recommender systems, their practical implementation has remained limited. In a recent systematic review conducted by da Silva et al. (2023) on educational recommender systems, it was observed that most studies suffer from a limitation in user sample size. This finding indicates that either these systems are predominantly utilized in restricted settings, or there is a lack of comprehensive reporting on their practical utilization.

2 Context and related work

Before diving into the details of the proposed framework, we clarify the context of the work and discuss related work.

2.1 Personalization granularity levels

Let us consider various personalization aspects within learning environments and their mapping to learning units of various granularity. Table 1 provides a simplified summary.

The most fine-grained learning units are simple unit tasks or individual steps within larger tasks. Examples of such learning units include a single vocabulary question, one step in solving an equation, and determining the correct spelling of a single word. Personalization at this level has been called "inner loop" (Vanlehn 2006) or "step loop" adaptation (Aleven et al. 2016); it typically takes the form of adaptive hints.

On the next level are tasks. Examples of tasks are solving an equation, checking and correcting the grammar of a sentence, or constructing a program in Python. Personalization on this level can take the form of task selection with the use of difficulty adjustment to achieve appropriate difficulty (Pelánek et al. 2017), choice of vocabulary for practice using spaced repetition algorithm (Settles and Meeder 2016), or personalized feedback on student's response (Maier and Klotz 2022). This type of personalization has been denoted as outer loop (Vanlehn 2006; VanLehn 2016) or task loop adaptation (Aleven et al. 2016).

On the next level in the granularity are learning activities. As a learning activity, we denote a situation when a student spends longer time intervals by learning a single topic (knowledge component) by either solving a series of related tasks (e.g., several equations with fractions) or studying materials related to the topic. Personalization on this level can take the form of mastery learning criteria, which decide when the student

has sufficiently mastered a particular topic and can move on Pelánek and Řihák (2018), Käser et al. (2016). Another personalization step is to recommend suitable follow-up activities or materials to study.

Finally, on a coarse level, we have whole lessons and courses, which take several hours to complete and which consist of many topics and many learning activities. Personalization on this level takes the form of course recommender systems (Gulzar et al. 2018).

The focus of this work is on the third level—learning activities—particularly with the focus on interactive activities where students solve some tasks or answers questions. In this setting, the goal is to recommend future activities.

2.2 Used terminology

As the terminology used to describe educational technology is far from standardized (Pelánek 2022), we explicitly clarify the terms used in this paper. We use the term *task* in a general meaning, which covers a wide range of interactions between students and learning environments: ranging from simple multiple-choice vocabulary questions to multi-step interactive problems in mathematics or programming. By *learning activity*, we denote a group of closely related tasks that are solved in a continuous sequence. *Topic* is some unit of knowledge.

Figure 1 provides an illustration of these notions and their relations. A learning activity consists of tasks and belongs under a topic. For a specific topic, we can have several different learning activities. Topics are typically organized in a taxonomy and connected with prerequisite or follow-up relations. Learning activities can be mapped to curricula (e.g., recommended grades). Some learning activities may have thematic motifs and may be mapped to student interests. We acknowledge that there are many approaches to the exact realization of domain modeling and that different learning environments use different terminology and decisions. For example, learning environments differ in the granularity of their domain modeling (topics); consequently, our distinction between a learning activity and a topic may be fuzzy in some settings. The core principles of the proposed framework are, however, not dependent on specific modeling choices.

2.3 Recommending learning materials

One direction of research for educational recommendations involves recommending general learning materials such as texts and videos or entire courses. This research considers the structure of the domain, including the relations between topics, prerequisites, and ontologies. However, it typically does not focus on analyzing students' answers or performance.

Manouselis et al. (2011, 2012) provide a general overview of recommender systems for learning. Raj and Renumol (2022) provide a recent systematic review of adaptive content recommenders. Another recent systematic review is provided by Rahayu et al. (2022) who focus specifically on "learning paths" (sequential recommendations of learning objects). Reviews of ontology-based recommendations are provided by

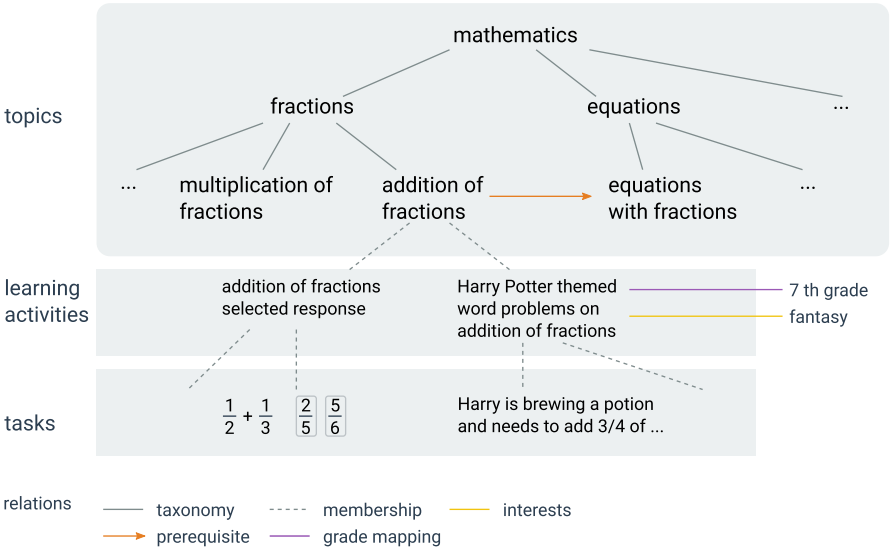

Fig. 1 Illustration of the used terminology using specific examples

George and Lal (2019), Tarus et al. (2018). An example of course recommender system is described by Gulzar et al. (2018).

2.4 Recommending practice

Another category of personalized educational techniques centers around interactive content that requires students to provide answers. These techniques incorporate student performance data and utilize it in student modeling methods to estimate skills. Using the student model as a basis, these techniques are capable of offering personalized recommendations.

These techniques are often applied on the level of tasks—a learning environment recommends (or even directly selects) the next tasks that are suitable for a student. A typical example is flashcard software, which focuses on learning factual knowledge like foreign language vocabulary; the focus of recommendations is on utilizing the spacing effect (Pavlik and Anderson 2008; Settles and Meeder 2016). Pelánek et al. (2017) describe a method for adaptive choice of tasks that focuses on utilizing estimates of prior knowledge. Arroyo et al. (2014) propose rule-based selection that takes into account student success and effort (e.g., response time, use of hints). Šimko et al. (2010) propose recommendations under a restricted time setting.

On the level of learning activities, personalization may concern mastery learning criteria and visualizations of progress toward mastery goals (Pelánek and Řihák 2018; Käser et al. 2016). This personalization element, however, does not involve recommendations.

Recommendation of practice is often closely interconnected with learning dashboards and methods for visualization of student knowledge estimates (open learner

models). Bodily and Verbert (2017) provide a review of learning analytics dashboards and educational recommendation systems.

Additionally, adaptive methodologies such as instructional planning (Vassileva 1995) and course sequencing (Brusilovsky and Vassileva 2003) predate the contemporary conceptualization of recommender systems. Although these works do not explicitly employ the term "recommendation," their objectives and employed techniques are closely related to recommendation of learning activities.

2.5 Closely relevant recommender systems topics

In the extensive body of literature on recommender systems, several threads have emerged that are particularly relevant to the recommendation of learning activities.

One such thread is *sequence-aware recommendations* (Quadrana et al. 2018). Unlike traditional recommender systems that focus on suggesting individual tasks, sequence-aware recommendations aim to recommend a cohesive sequence of tasks that work together well. Examples of such sequences include research papers (Mohseni et al. 2019) or tourism packages (Kolahkaj et al. 2020). Recommending learning activities may be viewed as a special case within the broader scope of sequence-aware recommendations, with many specific requirements given by pedagogical considerations.

Another closely related thread is *context-aware recommendations* (Adomavicius and Tuzhilin 2010). Contextual factors, such as time and location, are relevant across various recommender systems. Numerous techniques have been developed to incorporate these contextual factors, ranging from preprocessing and postprocessing methods to direct inclusion into recommendation algorithms.

A third notable thread is *explainable recommendations* (Zhang and Chen 2020). The primary objective here is to provide users with explanations that justify the specific recommendations they receive. Explanations play a crucial role in enhancing the persuasiveness and trustworthiness of recommendations (Tintarev and Masthoff 2010). Within the educational context, the significance of explainable recommendations is arguably even greater than in conventional recommender systems focused on movies or books. Barria-Pineda et al. (2021) offer a specific example of explainable recommendations within an educational context.

2.6 Rule-based recommendations

Our proposed approach is based on rules. Utilizing rules for recommendation offers several advantages. Firstly, rules are interpretable, which facilitates system development, debugging, and monitoring. Moreover, interpretability enables the provision of explanations for recommendations, enhancing transparency. Additionally, a rule-based system is relatively straightforward to extend or modify since individual rules can be changed or updated as needed. These advantages have not been overlooked in previous research, where various types of rule-based recommendations have been proposed.

One direction is based on the use of *association rules*, which is a standard technique in machine learning. Specific examples of their application in educational recommendations include learning object recommendations (Imran et al. 2016) and recommendations of learning lessons based on interests (Hsu 2008).

Another approach involves the use of *fuzzy logic rules*, which employ IF-THEN rules based on many-valued logic, allowing truth values of variables to range between 0 and 1. For specific examples of applications of fuzzy logic rules in educational recommender systems, see Asadi et al. (2019), Gogo et al. (2018). Rule-based systems, particularly those employing fuzzy logic rules, are often referred to as "expert systems." Such systems have been proposed, for example, for movie recommendations (Walek and Fojtik 2020) or diet recommendations (Tabassum et al. 2021).

While the existing research literature demonstrates the potential advantages of rule-based recommendations, most of the existing approaches are either limited in scope or specialized, making them unsuitable as a direct basis for the development of recommendations for learning activities in large-scale learning environments.

2.7 Evaluation of adaptive learning environments

Evaluation of recommendations (or adaptivity in general) in online learning environments is difficult (Weibelzahl 2005; Weibelzahl et al. 2020; Brusilovsky et al. 2004; Paramythis et al. 2010). First, it can take a long time for the effects on the students' learning and motivation to manifest. Furthermore, there is considerable noise and biases in the collected data, e.g., self-selection bias, mastery attrition bias, and ordering bias (Pelánek 2018).

The evaluation of adaptive learning environments can differ considerably. We can classify evaluations using the following four characteristics:

- *Summative vs. formative.* Summative evaluation can establish the effectiveness of the recommendation algorithm (or even the whole learning system); however, it does not indicate how to improve it. In contrast, formative evaluation provides insight into the behavior of the algorithm and how to improve it (Mark and Greer 1993).
- *Optimization vs. avoiding stupidity*. Optimization-aiming evaluation is based on metrics that describe the quality of the recommendations (e.g., click-through rate). Stupidity-avoiding evaluation attempts to detect "major inadequacies" (Mark and Greer 1993), such as ignoring prerequisites or recommending too difficult tasks leading to frustrating failures (Pelánek and Effenberger 2022).
- *End-to-end (whole system) vs. layered evaluation.* Layered evaluation refers to evaluating individual components in isolation, which is helpful for diagnosing what needs to change (Brusilovsky et al. 2004; Paramythis et al. 2010).
- *Long-term vs. short-term evaluation.* Long-term metrics are more closely linked to the aim of the algorithm but (obviously) take more time to measure and are not directly attributable to specific recommendations. Some examples of long-term metrics are mastered topics, post-test performance, results on standarized exams, and long-term retention. Some examples of short-term, online, attributable metrics

are the proportions of clicks, solved, solved with good performance, and solved if clicked (conditional probability).

These characteristics are not completely independent. The summative evaluation is typically optimization-focused, end-to-end, and long-term. The formative evaluation, in contrast, is often avoiding-stupidity-focused, layered, and shorter-term. For this work, more relevant is the formative evaluation since the purpose of evaluating recommendations in a large-scale learning environment is an iterative improvement.

Note that there are also other approaches to classify evaluations. For example, Iqbal et al. (1999) discuss experimental vs. exploratory and internal vs. external methods. Mark and Greer (1993) distinguish several evaluation methodologies: small group testing, field testing, and experimental research (which can be single group, control group, or quasi-experimental). Greer and Mark (2016) provide other examples of evaluation methods beyond controlled experiments, e.g., propensity-score matching, simulated learners, and comparing decisions made by adaptive learning environments and human experts.

3 Recommendation goals and requirements

To design a recommender system for learning activities, we need to clarify its goals and requirements. These goals and requirements naturally vary depending on the specific learning environment. For instance, the recommendations for vocabulary learning among elementary school students should differ from those required in vocational training. Our objective is to summarize the general goals and requirements relevant across a wide range of online learning environments.

3.1 General goals

On a high level, the general purpose of recommendations is to contribute to efficient student learning and engagement within a learning environment. We identify three key avenues through which this objective can be achieved: predicting intentions, facilitating exploration, and reinforcing knowledge.

3.1.1 Predicting intentions

The first objective is to efficiently guide students toward activities they would naturally be inclined to visit, even without the presence of recommendations. The purpose of these recommendations is not to surprise users but rather to enhance the efficiency and enjoyment of using the system. Additionally, they should increase the proportion of time students spend productively within the learning environment by guiding them toward the most suitable activities given their current knowledge.

This type of recommendation closely aligns with the concept of follow-up recommendations commonly employed in general recommender systems, such as those used for suggesting news stories.

3.1.2 Facilitating exploration

The second objective is guiding students toward content they might not actively seek out on their own. This could encompass alternative forms of practice than those previously mastered by a student, practice in different educational domains, unconventional activities like logic puzzles, or even content that is more advanced than their current grade level, provided the student previously demonstrated adequate knowledge to engage with it.

This type of recommendation aligns with the notion of *serendipity* often employed in general recommender systems. It aims to introduce students to unexpected yet valuable learning opportunities, expanding their horizons beyond their usual preferences and promoting a sense of discovery.

3.1.3 Reinforcing knowledge

The third objective is to provide recommendations that strengthen previously practiced knowledge, ideally targeting the most beneficial activities at any given moment. Personalized reinforcement of knowledge is one of the key potential advantages of computerized learning environments. However, engaging in repetition and knowledge strengthening is typically not something students are inclined to do without explicit impulses. The role of recommendations is thus particularly important.

The reinforcement may take different forms, such as repeated practice of previously learned concepts through spaced repetition, tackling the same topic using different interaction formats (e.g., shifting from selected response to constructed response), or practicing topics in an interleaved manner (e.g., integrating computation with fractions following separate practice of fraction addition and multiplication).

Recommendations for knowledge reinforcement are highly specific to learning environments and need to consider specific learning processes. As a result, this type of recommendation does not have a direct analogy within general recommender systems.

3.2 Recommendations situations

Within a learning environment, recommendations can be used in various places:

- *homepage recommendations* (learning dashboard)—a central location that the student visits when entering the learning environment (and potentially repeatedly during the system usage),
- *follow-up recommendations*—recommendations shown after a student finishes a learning activity,
- *default actions during navigation*—recommendations shown while a student is navigating the system (e.g., a user wants to practice operations with fractions; while browsing a tree of topics the system recommends a specific practice set concerning fractions),
- *special recommendations*—special pages that aim to motivate students toward targeted practice (e.g., reinforcing knowledge or toward a goal selected by a teacher); these may be connected with gamification elements.

Table 2 Mapping between recommendation situations and goals

	Predicting intentions	Facilitating exploration	Reinforcing knowledge
Homepage	✓ ✓	✓ ✓	✓ ✓
Follow-up	✓ ✓ ✓	✓	
Navigation defaults	✓ ✓ ✓	✓	✓
Special		✓ ✓	✓ ✓ ✓

Table 2 provides a rough mapping of recommendation situations and recommendation goals. The details, of course, depend on a particular setting and objectives of a particular learning environment. The main point, which should be relevant across various environments, is that different situations need to address different goals and thus need different recommendations.

3.3 Requirements for recommendations

For recommendations to truly benefit students, they must meet a range of specific requirements. The educational setting, in particular, gives rise to several unique demands not typically encountered in general recommender system contexts.

Recommend activities of appropriate difficulty. The basic requirement concerns appropriate difficulty—the recommended practice should not be too easy nor too difficult for the student. Determining appropriate difficulty is complex and involves the use of a domain model (taxonomy of topics, prerequisites, mapping to grade levels) and a student model (estimate of student skill based on past performance).

Support reinforcing and strengthening knowledge in a manner appropriate to the type of knowledge. Different types of knowledge require different types of learning processes (Koedinger et al. 2012) and may need to be reinforced in different manners. For example, for basic factual knowledge (foreign language vocabulary, names and locations in geography), it may be useful to recommend spaced repetition for the same practice that a student has already done. In contrast, for practicing programming, it is more useful to focus on the interleaving principle.

Take into consideration activities outside the learning environments. Computerized learning environments are typically used to complement other forms of learning. The recommendation algorithm thus needs to take into account that student knowledge state may have changed between practice sessions, particularly when there is a larger time window between them.

Take into account contextual information. Context awareness is a general trend in recommender systems (Adomavicius and Tuzhilin 2010). In learning environments, relevant contextual information includes: whether the activity is happening at school or at home, whether the activity was explicitly assigned by a teacher or parent (e.g., as homework), what type of device is used, whether audio output and input are available, current time and students time restrictions.

Supporting personalization with respect to interests. In addition to personalization based on student knowledge state, it is useful to take into account also student interests

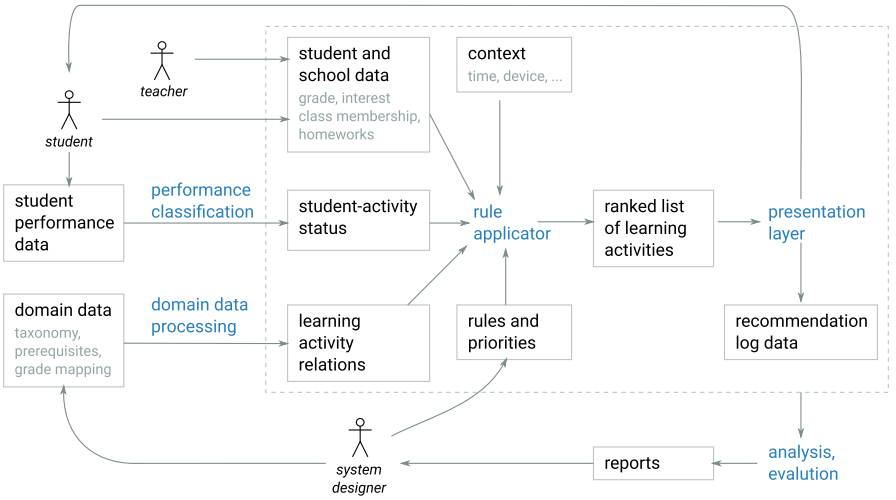

Fig. 2 Overall outline of the proposed modular framework

and to present tasks that are contextualized with respect to these interests (Cordova and Lepper 1996; Walkington 2013).

Provide diverse recommendations. Diversity is another generally important aspect of recommendations (Castells et al. 2021). Previous research in the educational setting shows that a suitable degree of novelty may sometimes be more important for motivation than difficulty factors (Lomas et al. 2017).

4 Rule-based recommendations

The proposed framework is based on a modular rule-based approach using IF-THEN rules. The individual rules are directly interpretable. The following simplified examples illustrate the basic form of the used rules:

- IF a student mastered an activity A without problems and B is an activity for a follow-up topic THEN recommend B.
- IF a student struggled while practicing activity A and it has been more than 10 days since the last practice THEN recommend A.

The individual rules are relatively simple. The focus of the framework is on properly managing the data required by the rules (student states, activity relations, contextual data) and developing a ruleset that well captures the requirements on recommendations.

4.1 Overview of the proposed framework

Figure 2 illustrates the overall structure of the proposed framework. We start by outlining the main role of individual parts. In the rest of the section, we discuss the modules in more detail.

The input to the recommendation process consists of the following data:

- *student and school data*: data specified by the student or teacher, e.g., students' class grade, class memberships, interests, specific homeworks and their deadlines,
- *student performance data*: data about previous student activity within the learning environment (e.g., correctness of answers, response times),
- *context*: automatically determined contextual information, e.g., time of day, device used by a student, activity of classmates,
- *domain data*: meta-data concerning the educational content, specified by system designers, e.g., mapping of learning activities to topics, prerequisites among topics, mapping of learning activities to grades,
- *rules and priorities*: data specifying the behavior of the recommender system, specified by system designers.

The input data are processed in several steps, which produce intermediate data (all intermediate data are interpretable). The *performance classification* processes raw data about student activity into abstracted *student-activity status* data, e.g., based on the correctness and response times of Jane's responses to a series of vegetable vocabulary questions, the system may store the status "Jane, vegetable vocabulary, mastered well." Another preprocessing of input data involves domain data, where *learning activity relations* are automatically computed based on high-level content meta-data.

The core of the algorithm lies in the *rule application*, which applies the specified rules to student data. Although this step combines many data sources, its realization is quite straightforward—a stepwise evaluation of individual IF-THEN rules, where each of these rules references only some data sources. This step produces a *ranked list of learning activities* for recommendation.

The *presentation layer* takes the computed list and presents it to the student while taking into account the goals of different presentation locations, as outlined in Table 2. The data about presented sets are also stored as logs.

Finally, an inherent part of the framework is *analysis and evaluation*. This step takes the stored logs and creates reports that are used by the system designer to tune the rules and domain data. The framework thus supports continuous iterative improvement.

4.2 The rationale for the chosen approach

We take the avoiding stupidity perspective (Pelánek and Effenberger 2022)—rather than aiming at optimal recommendations, our main goal is to avoid undesirable behavior. As listed above, the recommendations try to achieve several goals and need to take into account multiple requirements. It is hard to satisfy perfectly even a single of these requirements—research papers often go into depth into one of them (e.g., determining what is appropriate difficulty or optimizing spaced practice intervals), and even with such clear focus, it is hard to find optimal solutions. For practical application in a learning environment, we do not necessarily need optimality, but it is essential to have reasonable coverage of all requirements. It is not useful to have a sophisticated model of prerequisites but completely ignore the issue of forgetting.

The proposed approach thus aims to provide reasonable coverage of all these requirements and to be applicable in a real-life setting of a large-scale learning environment. With this goal in mind, the modular rule-based framework has several advantages:

- *Modularity.* Individual components can be developed, tested, and evaluated separately.
- *Extensibility.* The recommendation system can be improved incrementally by adding new rules.
- *Flexibility.* The framework is applicable for recommendations in a wide range of educational domains.
- *Explainability.* For each recommendation, we can determine the reason why it is recommended. This is very useful for the development (verification, monitoring, and evaluation of the system) and may also be applied in the user interface—providing students explanations for recommendations, which may be useful for building user trust (Tintarev and Masthoff 2010).

For a comparison, consider an alternative "black-box approach," where we use machine learning techniques (neural networks, reinforcement learning) to learn recommendations directly from data. This would save time needed by human experts (no need to write recommendation rules or tune parameter values), but it would lack the above-mentioned advantages.

4.3 Student performance classification

With respect to the use of student performance data, the core idea used in the framework is to use abstraction of the observed performance rather than estimates of latent skills.

Student modeling research primarily focuses on the estimation of student cognitive skills (Pelánek 2017). However, when it comes to real-world, large-scale learning environments, applying the student models outlined in research papers becomes intricate. These models require fine-grained skill models (task-skill mappings), which are time-consuming to prepare and manage. In scenarios where multiple skills are involved per task, determining how to assign credit or blame becomes unclear. Additionally, there is a lack of research on utilizing student models in cases where multiple activity types exist for the same topic. For example, consider the practice of vocabulary using multiple-choice questions and written answers. Should these activities be treated as practicing the same skill or rather distinct yet somehow related skills? This aspect is not sufficiently explored in current research. Furthermore, the models that try to address these more complex modeling issues are often computationally demanding and are not readily applicable in practical settings that require frequent skill updates.

Accurate skill estimates are particularly valuable when it comes to adapting learning experiences at the step and task levels. These estimates enable personalized hints, feedback, and dynamic difficulty adjustments within a learning activity (as outlined in Table 1). However, when it comes to recommendations, we adopt a different approach. Instead of estimating latent skill, we use abstraction of the observed performance and we base the recommendations directly on this abstracted performance. Rather than delving into the complexities of detailed student modeling, our emphasis lies in

Table 3 Student performance classification: example of used status classes and the basic idea of classification criteria

Classification	Criteria
Easy mastery	Low error rate and low time to mastery
Weak mastery	High error rate or high time to mastery
Normal mastery	Other cases of mastery
Wheelspinning	Many attempts, mastery not reached
Tried	Mastery not reached, not wheelspinning

combining student performance data with other pertinent factors such as contextual information and student preferences.

For each learning activity that a student takes, we abstract the observed performance data (correctness of answers, response times) into a discrete performance classification. Table 3 illustrates the basic idea of such classification. (The choice of classes and details of criteria depend on a particular learning environment.) This approach is analogical to the one proposed in Pelánek and Effenberger (2020), where the classification approach was used for individual responses. Here, we apply it on the level of learning activities.

4.4 Domain data for recommendations

The recommendation rules are based on domain data, particularly various relations and attributes of topics and learning activities (see Fig. 1 for illustrations):

- generalization-specialization relationships, which organize the domain into a taxonomy,
- prerequisites, typical follow-ups or paths through topics, which specify a suitable ordering of activities,
- mapping between learning activities and topics,
- mapping between learning activities and interests,
- mapping of topics or learning activities to grades for which the content is suitable,
- difficulty rating of learning activities.

The relations can be specified in different ways: manual specification by an expert (e.g., using curricular documents), automatic computation based on high-level domain data (e.g., using some manually specified relations to compute automatically derived ones), or automatic computation based on student activity within the environment.

4.5 Rules

Rules are parametrized by learning activities and have the form of IF-THEN rules: *IF condition(A, context) THEN recommend A*. The conditions may refer student performance classification over activities, domain data relations, learning activity features, or various contextual information (e.g., timing, device). Each rule also has assigned a priority; these are used to select and rank applicable recommendations.

Table 4 Examples of rules for recommending learning activity A to a student s

Rule name	Condition	Priority
Follow topic	s mastered B well, $(B, A) \in$ follow	0.9
Pred topic	s wheelspinning B, $(A, B) \in$ follow	0.8
Repetition normal	s mastered A normally, at least 10 days ago	0.5
Homework	s has homework A and current time $>$ 2PM	1

Table 4 shows illustrative examples of rules (simplified). Specific rules depend on the details of a particular learning environment. There are, however, several broad classes of rules that are useful across learning environments. These classes are useful for grouping recommendations in the user interface.

4.5.1 Follow-up activities

If a student successfully finished a learning activity, we want to recommend a follow-up activity that somehow extends or builds upon the completed one.

Individual rules capture different types of follow-ups:

- difficulty, i.e., recommending the same type of activity, the same topic, but a more difficult activity,
- activity type, e.g., after a student finishes a selected response activity for vegetable vocabulary, recommend vegetable vocabulary with written answers.
- topic,
- presentation ordering within a system.

The priority of rules may depend on student performance. For example, the follow-up with respect to difficulty may have high priority only when a student shows fluent mastery within the current activity.

4.5.2 Preceding activities

If a student does not finish an activity even after many attempts or finishes the activity but is struggling, we want to recommend activities that are "preceding" and will help the student to better master the topic he is struggling with.

These rules closely resemble the follow-up rules, but their direction with respect to domain relations is reversed.

4.5.3 Strengthening

We want to recommend activities that strengthen students' mastery of topics that they have practiced previously. Efficient methods for such strengthening are distributed practice (spaced repetition) and interleaved practice, which have been shown to robustly improve learning across various learning domains (Carpenter 2014; Dunlosky et al. 2013).

The spaced repetition strategy leads to the following type of rule. If a student finishes a learning activity, recommend the same activity after a suitable time delay. The priority of the rule should be based on the time delay and the current student performance. (Priority should be given to the repetition of topics where the student struggles.)

The interleaved practice strategy leads to rules of the following type. If a student finishes learning activities for topics A and B, recommend an activity in which topics A and B are interleaved.

4.5.4 Exploration

We want also to navigate students to activities that may be relevant for them but are not directly connected to their previous activity. This is particularly useful for new users in the learning environment as they have limited recorded activity.

Specific examples of such exploration rules:

- featured activities—manually or semi-automatically selected activities,
- interests—recommendations based on student's selected interests, e.g., recommending activities featuring animals or movies,
- peer activity—recommendation based on the recent activity of peers, e.g., classmates.

4.5.5 Assigned activities

Recommendations also need to take into account various features of learning environments which lead to explicitly assigned activities. A typical example is the support for assigning homework to students by their teacher or parent.

4.6 Computing recommendation candidates

Once we have the recommendation rules and the underlying domain and student data, we use them to generate a list of potential recommendations. By design, the application of these rules is straightforward. For each rule, we evaluate its condition, and if the condition is met, the corresponding activity is appended to the recommendation candidate list along with its priority and the name of the rule that generated it.

The process of computing recommendations entails addressing a few technical considerations. For instance, some filtering aspects are relevant across all rules. For example, it is important to recommend only those activities that align with the student's grade level or are compatible with the device they are currently using. Rather than including these filters directly within the rule conditions, it may be more appropriate to apply them as separate postprocessing steps on the candidate list.

Another technical aspect relates to computational efficiency. One approach is to precompute the list of candidates for each student through batch precomputation once a day. Throughout the day, as students complete learning activities, the candidate list can be updated based on the rules that are directly relevant to the outcome of a given activity.

4.7 Presenting recommendations

Finally, once we have identified recommendation candidates, the last step is to present them to students. As discussed previously and illustrated in Table 2, the relevant type of recommendation varies depending on the recommendation situation.

To realize the presentation, we have to make the following decisions:

- Determining which rule types should be employed for each recommendation situation.
- Establishing the number of recommendations to display in each situation.
- Selecting a method for choosing among the candidates. The method should consider recommendation priorities and try to achieve diversification. This can be done, for example, by utilizing a variation of roulette wheel selection.
- Designing the specific layout and appearance of the recommendations. Each final recommendation has a clear reason behind it (specific rule, rule type, seed learning activity). These reasons can be used in the user interface to structure recommendations and provide explanations (Tintarev and Masthoff 2010).

5 Monitoring and evaluation of recommendations

In this section, we discuss tools for monitoring and analysis of recommendation behavior. Using the characteristics of evaluation methods from Sect. 2.7, we focus on formative, stupidity-avoiding, short-term evaluation, which is well-suited to help the system designers to iteratively improve the system.

5.1 Evaluation of modules

A key advantage of a modular framework is that it allows layered evaluation of the individual modules. Figure 3 shows the four modules that can be evaluated in isolation: 1. performance classification, 2. learning activity relations, 3. recommendation rules application, and 4. presentation layer.

5.1.1 Evaluation of performance classification

The performance classification module takes raw student performance data and produces student statuses (Sect. 4.3). The goal of evaluation is to check the validity of these statuses and provide insights for improvement. Typical actions based on the evaluation are tuning the thresholds or criteria used for classifying the performance.

The evaluation can be performed using several approaches. The basic check is provided by simple descriptive statistics like the distribution of student statuses into individual classes (Pelánek and Effenberger 2020). If the distribution is highly uneven (dominated by a single class), then the information cannot be very useful for personalization.

A detailed insight, although with a limited scope, can be provided by an expert in-depth inspection for a sample of data. The expert is provided with a readable listing

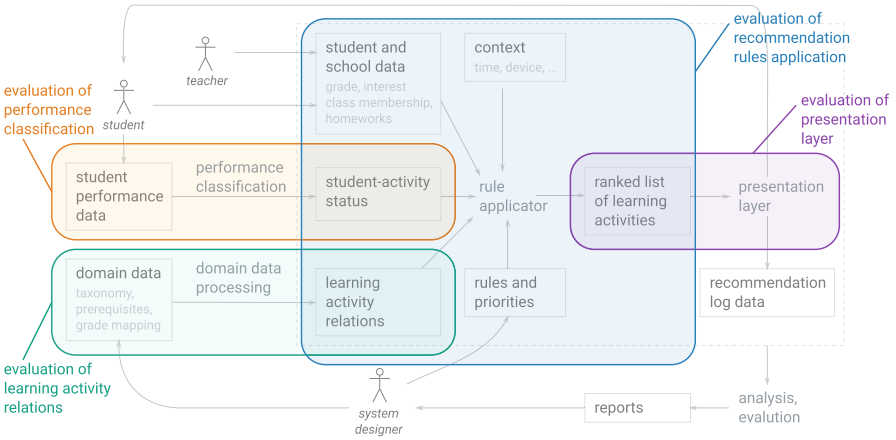

Fig. 3 Evaluation of individual modules displayed as an overlay over the framework structure (Fig. 2)

of the raw data for sample student attempts and provides a judgment on the performance statuses for given attempts. This judgment is then compared with the outputs of the classifier. The specific procedure can vary and can be found in the literature under various names, such as expert-based evaluation (Mark and Greer 1993), expert inspection, on-site expert evaluation, panel of experts (Iqbal et al. 1999), or heuristic evaluation by experts (Paramythis et al. 2010).

A fully automated evaluation of the quality of statuses can be performed by psychometric techniques for assessing validity of measurements (DeVon et al. 2007). Predictive ability of the statuses can be quantified using accuracy of the next answer predictions based on status information (Pelánek and Effenberger 2020). We can even analyze the impact of the performance measure on the validity and reliability of student and domain models that use the performance statuses as their input. For example, Effenberger and Pelánek (2021) compare multiple performance measures using validity and reliability of skill and difficulty estimates computed using the performance statuses.

5.1.2 Evaluation of learning activity relations

The recommendation rules can refer to various domain data (Sect. 4.4), which can be evaluated separately. Each type of domain data may require slightly different evaluation; we will give examples of relations between learning activities.

No matter whether the relations are specified manually or computed automatically from other domain data, it is useful to check the final relations. The basic check is again provided by descriptive statistics, e.g., summary statistics like the density of relations and listings of the most and least connected nodes. These statistics are useful particularly for identifying weak spots due to some clear mistake or omission. A deeper analysis needs to involve a human expert, who performs an in-depth check of a sample of computed results.

5.1.3 Evaluation of recommendation rules and their application

This evaluation is concerned with rules and their application (Sect. 4.5). For this step, we assume that the student statuses and learning activity relations are correct, and we check the recommendations (and their priorities) produced by the rules.

A basic approach is an evaluation using human experts, who check the outputs for provided inputs. The inputs consist of the student-activity statuses, learning activity relations, student and school data, as well as the context (Fig. 3). These inputs can be specified directly, synthesized using simulated users (Paramythis et al. 2010; Greer and Mark 2016), or by sampling the existing users. The output consists of all activated recommendation rules and produced recommendations, together with their priorities.

These input–output pairs are then checked by a human: Assuming the correctness of the student-activity statuses and of the learning activity relations, are the recommendations appropriate for the given student in the given context? Should something else be recommended, e.g., is there a "strengthening" recommendation that a human teacher would suggest? Typical actions are updating priorities, modifying the rules, or even adding new rules.

It might be hard for human experts to take into account the extensive domain data and to assume their correctness. Instead, they may rely on their domain knowledge (Paramythis et al. 2010). In this case, the evaluation is not completely isolated and the human experts might suggest changing the domain model to fix inappropriate or missing recommendations—and it might, indeed, be a more suitable remedy than changing the rules.

We want not just to tune the rules that we already have but also to discover "missing rules" that would be valuable to add. Using user data, we can extract the events in which a user solves (with an appropriate performance) a learning activity that was not recommended to her. Alternatively, as an even stronger indication of a suitable missing recommendation, we can extract events when a teacher assigned a class an activity that would not be recommended to the pupils using the current rules.

5.1.4 Evaluation of presentation layer

Based on pre-computed data (activities to recommend, priorities, responsible rules) and context (recommendation situation, time), this module presents the final recommendations (Sect. 4.7).

We can employ human expert evaluation, as in the previous section. The key difference is that now we assume the correctness of the pre-computed list of recommended activities with their priorities—which is the input that should be shown to the human expert. Again, we can either specify the inputs directly or use the collected data for a representative sample of users. We ask the experts if the presentation (e.g., selection, grouping, ordering) is appropriate. As in the previous layer, however, it might be difficult for the experts to assess only the presentation decisions, if they do not think the list of recommendations is appropriate (Paramythis et al. 2010).

Table 5 Metrics to quantify the performance of the recommendations

Metric	Definition	Label/calculation
Displays/day	How many times displayed	N
Clicks/day	How many times clicked on	C
Solved/day	How many times clicked on and solved	S
Click-through rate	P(clicked | displayed)	C/N
Conditional success rate	P(success | clicked)	S/C
Success rate	P(success | displayed)	S/N

5.2 Evaluation metrics for recommendations

During the layered evaluation, we evaluated each module in isolation. It is, nevertheless, also important to check that the modules work well together and the system as a whole. Since we are interested in formative evaluation and iterative improvement, we focus primarily on metrics that are directly attributable to actions of the system (Zinkevich 2017). Although these metrics are only proxy measures for our real goals (e.g., efficient student learning), they can be realistically measured and can provide valuable insights for system improvement.

Table 5 lists six basic metrics that are attributable to individual recommendations or recommendation batches.[1] We can look at both the counts (absolute frequencies) and rates (relative frequencies) of the desirable events—recommendations being displayed, clicked on, and the recommended activity being solved by the user. The basic counts are the number of displayed recommendations, how many times a recommendation was clicked on, and how many times a recommended activity was successfully completed. For better interpretability, the counts can be averaged per a meaningful unit of time (e.g., clicks per day).

When a batch of recommendations is shown, we want the user to click on (and solve) one of the recommended activities, not all of them. We can count the same three events aggregated per batch: the total number of batches displayed, how many times at least one recommendation in the batch was clicked on, and how many times at least one recommended activity in the batch was clicked on and successfully completed.

Using these counts, we can calculate three different rates (relative frequencies): click-through rate (ratio of clicks to displays), conditional success rate (ratio of success to clicks), and (unconditional) success rate (ratio of success to displays, product of the former two rates). All these rates can be measured either per a recommendation or per a batch of recommendations.

As we will see in the case study (Sect. 6.3), none of these metrics is sufficient to look at alone. For example, high rates are not meaningful if the counts are very low. Similarly, a high click-through rate is not something to celebrate if the conditional success rate is low.

[1] A *recommendation batch* is a group of recommendations shown to the user at once, e.g., all recommendations the user gets on the homepage.

5.3 Comparison of rules across context

It may be hard to interpret the above-listed metrics in isolation, e.g., it is not clear what a "good" click-through rate is. By comparing results for individual rules and by monitoring the user interactions with the recommendations in various contexts (e.g., homepage vs. next to solve, English vs. Math, primary vs. secondary school), we can more easily detect problems, e.g., underused rules (which may be wrongly specified), or identify leverage points, e.g., heavily used rules, for which small changes can have a large impact and it is thus worthwhile to pay attention to details of their formulation and prioritization.

Typical actions based on the evaluation include an update of priorities, a change of a rule, or even an addition or removal of a rule. However, compared to the evaluation of modules, end-to-end evaluation gives less guidance on what action to take if an issue is detected—any module may be the culprit. Sometimes, a discrepancy between various slices suggests what to change. For example, if the performance is poor in a single educational subject, it may be because of insufficiently specified domain data. Other times, a follow-up evaluation of the modules is necessary to pinpoint the root cause of the issue.

The metrics discussed in Sect. 5.2 can be computed for the whole system or for various slices of the data, such as the educational subjects, subpopulations of users (e.g., school grade), or recommendation situations (e.g., homepage, next to solve). The metrics for the individual recommendations can also be computed per each recommendation rule, which allows us to evaluate the performance of each rule separately. We will illustrate such an evaluation in a case study (Sect. 6.3).

Note, however, that it is tricky to compare the performance of the rules since the rule to generate a specific recommendation is not selected at random. There are many influencing contextual factors impacting the metrics, such as the number of displayed alternatives and the position of the recommendation within the batch. Some of these factors can be partially accounted for in the analysis or in the data collection by randomizing some choices (basically performing an online AB experiment to compare the rules).

6 Case study: Umíme learning environment

In this section, we report on the implementation of the proposed framework in the learning environment Umíme. The implementation covers all aspects of the framework discussed in the paper and is used in the production version of the environment.

In the discussion, we focus particularly on the illustration of evaluation methods described in Sect. 5. Our goal is not to make claims about the quality of the specific implementation of the framework or specific recommendation rules—such claims would not transfer to other contexts anyway. Rather, our goal is to demonstrate the utility of the formative, layered evaluation methods, which are enabled by the interpretability and modularity of the proposed framework. Section 6.3 illustrates several quantitative evaluation methods and highlights specific caveats of the evaluation. Sec-

tion 6.4 presents a high-level summary of actionable insights and practical results of the layered evaluation that we obtained during the practical application.

6.1 Scope and usage

Umíme is a learning environment focused on adaptive practice with a wide coverage of subjects, including native language grammar and orthography, second language learning, mathematics, programming, biology, and geography. It includes thousands of learning activities with multiple interaction and assessment types, e.g., multiple-choice questions with mastery criteria or interactive programming exercises with automated testing. In addition to recommendations, the system also uses adaptation in the form of mastery criteria; the specific algorithm is described by Pelánek and Řihák (2018).

The environment is available primarily in the Czech language, and the majority of users are from the Czech Republic. Slovak and Polish localizations are also available. The primary target audience is elementary and high school students. The environment can be used freely with a limited number of daily answers; full access is available with school and individual licenses. The environment is used by tens of thousands of students per day. Pelánek (2021) reports more details about the usage of the environment.

Within this learning environment, we fully implemented the recommendation framework described in this paper; it has been applied in the production version of the environment since spring 2023. Given the wide coverage of educational subjects within the learning environment, the case study practically demonstrates the wide applicability of the rule-based approach.

6.2 Domain data

The basic organization of domain data within the environment corresponds to Fig. 1: a taxonomy of topics, with learning activities mapped to topics. For the determination of follow-up and precedence relations, we use the following combined approach:

- relations among topics are manually specified based on recommended curricula and human expertise,
- learning activities are mapped to topics and have several attributes (difficulty rank, recommended grade) which are specified manually and refined using data-driven heuristics with human-in-the-loop,
- relations between learning activities are computed automatically using simple interpretable rules based on the manually specified data.

The details are slightly more complex due to the fact that the system uses different types of assessment criteria—mastery criteria for sequences of simple tasks (e.g., grammar, vocabulary learning, simple computations) versus assessment of individual, larger tasks (e.g., in programming or reading comprehension).

Table 6 Metrics for a selection of recommendation rules

rule	displays per day	clicks per day	solved per day	CTR	CSR	SR
`follow-up-diffRank`	11000	220	57	2.1	26	0.5
`follow-up-kc`	12000	170	45	1.4	27	0.4
`pred-for-weak-kc`	2900	26	7	0.9	26	0.2
`pred-for-weak-diffRank`	840	9	4	1.1	39	0.4
`repetition-for-weak`	5300	28	11	0.5	40	0.2
`repetition-for-normal`	15000	76	38	0.5	50	0.3
`homework-follow`	790	9	5	1.2	50	0.6
`homework-deadline`	3600	65	34	1.8	53	1.0
`featured`	150000	1500	300	1.0	20	0.2
`peers`	11000	120	51	1.1	42	0.4

The metrics are defined in Table 5 (CTR = click-through rate, CSR = conditional success rate, SR = success rate). The counts are rounded to two significant digits. The rates are expressed as percentages. Highlighted values are the maximum and minimum in each column

6.3 Caveats of recommendation rules evaluation

Each recommendation rule can be thought of as a simple recommendation algorithm, so it is natural to compare them. Which rule works best? Which rules should be dropped? However, comparing rules based on end-to-end evaluation without a proper randomized controlled experiment can lead to misleading conclusions. Let us illustrate this point.

For this illustration, we report results for three subjects (English as a second language, Mathematics, Computer Science), three recommendation situations (Homepage, Next to solve, Navigation in the exercise dashboard), and 10 diverse rules that cover various recommendation goals (predicting intentions, strengthening knowledge, supporting exploration). The data were collected during 11 weeks in spring 2023 and consists of 16 million recommendations (208 thousand per day), which were displayed in 3.1 million recommendation batches (40 thousand per day).

Table 6 compares the overall performance of a selection of rules. Note that to compare rules, we can only use the per-recommendation and not the per-batch metrics (defined in Sect. 5.2) since each batch contains recommendations generated by multiple rules.

First, the results highlight the importance of reporting multiple metrics: the ranking of the rules differs a lot depending on the chosen metric. In particular, the rule with the highest counts (`featured`) has the lowest success rate (both conditional and unconditional). The rules also behave differently with respect to the various rates. For example, the rule with the highest click-through rate (`follow-up-diffRank`) has one of the lowest conditional success rates.

Second, the usefulness of different rules varies across contexts; it is not possible to give a clear-cut evaluation of which rule is the best. Table 7 illustrates this point for a selection of rules, situations, subjects, and populations of users. For brevity, it

Table 7 Click-through rates for a selection of rules across a selection of situations, subjects, and populations. The background color corresponds to the CTR in the cell; the darker the color, the higher the CTR

	situation			subject			grade		
	Home	Next	Navig.	Eng.	Math	CS	unset	1–9	10+
follow-up-diffRank	1.2	3.1	3.5	2.7	2.0	1.1	2.9	1.4	1.3
follow-up-kc	1.1	1.6	2.5	1.9	1.3	0.7	1.7	1.1	1.1
pred-for-weak-kc	0.8	0.7	2.1	1.1	0.8	0.4	1.1	0.7	0.9
pred-for-weak-diffRank	1.2	0.8	2.0	1.4	0.9	0.7	1.3	0.9	0.6
repetition-for-weak	0.5	0.6	2.3	0.8	0.5	0.3	0.8	0.4	0.7
repetition-for-normal	0.4	0.6	1.7	0.8	0.5	0.3	0.7	0.4	0.8
homework-follow	1.2	5.1	0.0	0.7	1.0	1.8	2.0	1.1	1.0
homework-deadline	1.8	0.0	0.0	1.1	1.8	2.3	3.2	1.6	1.6
featured	0.7	0.6	4.1	1.1	1.0	0.9	0.8	1.6	1.2
peers	1.0	0.8	2.2	1.1	0.7	1.6	1.6	1.0	0.6

Recommendation situations: Homepage, Next to solve, Navigation in the exercise dashboard. Subjects: English, Mathematics, Computer Science. Grades: unset, primary and secondary school (grades 1–9), high school, and older (grade 10+)

only shows the click-through rate (CTR). As shown in Table 6, other metrics would give not just different values but even different ordering of the rules. Nevertheless, the message holds for all metrics we discussed: both the specific values and the ordering of the rules depend on the context (situation, subject, and population).

Global averages can be misleading. Some rules are used in some contexts more frequently and some contexts are more favorable to some metrics. For example, there are more recommendations on the homepage than on the next to solve page, so the rules that are used (relatively) more frequently on the next to solve page tend to have higher click-through rates.

It is even possible for the comparison with respect to the global average to differ from the comparison with respect to the average of per-context averages. Consider, for instance, a comparison of `follow-up-kc` and `homework-follow` rules. The first rule has a higher global click-through rate (1.4 vs. 1.2), while the second rule has a higher average of per-situation click-through rates (2.1 vs. 1.7). How can this happen? Both rules have similar click-through rates on the homepage (1.1 vs. 1.2), but while the `follow-up-kc` is frequently used in other situations, the `homework-follow` is rarely used elsewhere. The global average is thus more heavily influenced by this unfavorable context for the `homework-follow` than for the `follow-up-kc` rule.

Such inversion of the comparison is not uncommon: for the recommendation situation, this happens in 13 out of 45 possible rule pairs, for subjects in 2 pairs, and for grades in 6 pairs. We have even observed instances of the Simpson paradox (Kievit et al. 2013): the global average of one rule being greater than that of another rule, despite the other rule having better results in *all* subgroups.

Figure 4 visualizes one instance of the Simpson paradox. The rule `follow-up-diffRank` has a higher global click-through rate than the rule `homework-`

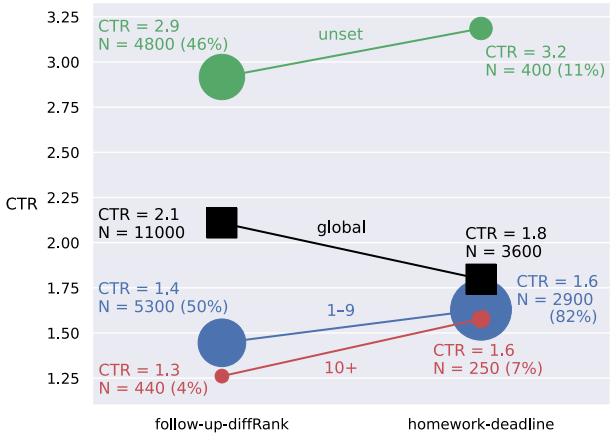

Fig. 4 Simpson paradox: the `follow-up-diffRank` rule has a higher global click-through rate, although it has a lower click-through rate in all subgroups defined by user grade. All values are per-day averages. The size of the circles reflects the within-rule proportion of displays from which the click-through rate is computed (i.e., the weight for the computation of the global average)

`deadline` (2.1 vs. 1.8, see Table 6). It has, however, lower click-through rates in all subgroups defined by user grade (see Table 7). Understanding why this happens gives us insights into the interaction between the users and the recommendation algorithm.

First, observe that CTR is notably higher for users with an unset grade. One reason might be that since we have less information about these users (not just the missing grade, but typically also just a short history of activity in the system), we give them fewer recommendations. This is especially true on the homepage, where the users with an unset grade get 7 recommendations on average, compared to 11 for the users with a known grade. The users' clicks are diluted between fewer recommendations, so the click-through rate is higher.

Second, the rule `homework-deadline` is relatively rarely used for users with an unset grade. This is not surprising since homework is given to pupils in a class, and those typically have a known grade. Only 11% of all recommendations produced by this rule are for users with an unset grade, compared to 46% for `follow-up-diffRank`. When we compute the global average, `follow-up-diffRank` thus benefits much more from the high CTR of the users with an unset grade, and this benefit overweights the fact that it has a lower CTR across all subgroups.

To summarize this section, the evaluation of rules is complex, even if we focus on their immediate impact on student activity. As we have seen, one should avoid premature conclusions about one rule working better than another based on just global averages. There are many confounding variables, e.g., how many other recommendations were displayed at the same time, the ordering of the recommendations, and the population of users. Moreover, the details of measurement matter too—we have seen the difference between various metrics, but even for a single metric, there are multiple reasonable ways to measure them (e.g., the "finished activity" in the definition of the conditional success rate can be defined in several ways).

6.4 Iterative improvement based on layered evaluation

During the development and application of the algorithm, we repeatedly performed evaluation steps corresponding the layered evaluation described in Sect. 5. Here, we briefly describe the main actionable results of the evaluation. The basic types of actions were the following: tuning of algorithm parameters, modification of rules (e.g., adding new antecedents to conditions), the removal and addition of rules. The analysis performed within layered evaluation also helped to uncover several implementation defects, which would be hard to find using standard software engineering approaches (due to lack of ground truth, standard testing methods are hard to apply).

In the evaluation of *performance classification*, we analyzed distributions of student-activity statuses (Sect. 4.3) across exercises (and other slices of data). Based on these results, we calibrated the thresholds to avoid too skewed distributions (e.g., no easy or weak mastery). We have also found a few bugs causing incorrect or missing measurements in some contexts.

In the evaluation of *domain data processing*, we checked the automatically computed relations between learning activities described in Sect. 4.4. Distribution analysis revealed some cases when the high-level, manually specified relations led to too many specific relationships between the activities, and we narrowed them down. We have also found some missing expected relationships, which were caused by missing or misspecified high-level relationships.

In the evaluation of *rules*, even the most basic metrics, such as the number of displays (Sect. 5.2), revealed several issues and potentials for improvements, such as:

- two rules with very few displays due to implementation bugs which caused them not to be applied in some contexts,
- a successful rule (to continue an unfinished learning activity) that only applied to activities with a specific assessment type; we then added an analogical rule for another assessment type,
- repetition rules being used rarely due to low priorities that dropped to zero after some time; we then changed the function that assigns them priority based on the time since the last mastery.

We have also analyzed frequent batch sizes and compositions. We found frequent batches with a single (or very few) recommendations on the homepage in some subjects (history, physics), especially for new users. This was addressed by extending the set of activities marked as featured.

In the evaluation of the *presentation layer*, we found unexpected results in the case of the "Next to solve" recommendation for homework activities. In this case, the presentation was configured in such a way as to display primarily other homework activities; we expected these to be very natural recommendations that would be used a lot by students. However, the data showed very minor usage. Our hindsight interpretation is that after finishing one activity within a homework, students want to see an overview of the homework to get a better sense of progress (rather than directly jumping into another activity as the system recommended).

7 Summary

In this work, we propose a specific approach to recommending learning activities. In this final section, we highlight the core ideas behind this proposal and outline directions for further research in this area.

7.1 Recommendations at learning activity granularity

Personalization in computerized learning environments can be considered at various levels of granularity, from adaptive hints within single steps in equation solving to recommending whole courses. The presented recommendation framework addresses recommendation at the level of learning activities, i.e., homogeneous sequences of tasks that typically take a few minutes to finish. We argue that this is a practically important problem that did not get sufficient research attention so far.

Recommendations at this granularity level are complex, as they require us to take into account several different aspects, including prerequisites, suitable difficulty, strengthening of knowledge, forgetting, student interests, and homework.

7.2 Avoiding stupidity perspective

It is tempting to approach the design of recommendations as an optimization problem. This perspective may be feasible in certain contexts, such as product recommendation, where we may want to optimize sales.[2] In the education setting, this perspective is not reasonable. Learning activities involve multiple objectives, including short-term learning, long-term knowledge strengthening, and fostering engagement. Moreover, these objectives are hard to measure. This makes the optimization perspective impractical.

Since recommending appropriate learning activities is complex, many things can go wrong. Thus, rather than optimizing recommendation, we argue for the use of the "avoiding stupidity" perspective (Pelánek and Effenberger 2022; Baker 2016; Mian et al. 2019; McNee et al. 2006), i.e., using an approach that allows us to efficiently develop a recommender system without clear shortcomings and which supports continuous monitoring for weak spots and their iterative improvement.

7.3 Interpretable rule-based approach

The proposed framework is based on interpretable IF-THEN rules that directly correspond to the pedagogical rationale behind the recommendation; this is in stark contrast to black-box approaches to recommendations based on neural networks or reinforcement learning.

This approach has several advantages:

– *Ease of application and design scalability.* It is possible to apply the framework with a small rule set rules and then gradually expand it. This makes the framework

[2] Even there, it would be short-sighted to optimize only short-term sales, which could undermine user trust and consequently long-term sales.

applicable also for learning environments with a small budget, e.g., those targeting education in smaller countries where, compared to the English language setting, it is harder to get funds for the development of personalized learning environments.
- *Computational scalability.* The framework is based on a collection of rules that can be efficiently evaluated. The approach is thus easily applicable in large-scale learning environments.
- *Transparency.* It is possible to give users an explanation of a recommendation.
- *Controllability.* System designers can easily control what gets recommended to students by (de)activating individual rules or changing their priorities. It would be possible to extend the framework to give control also to teachers or students.

Compared to black-box machine learning methods, the disadvantage of the approach is that it requires designers to create the complete rule set for recommendations.

7.4 Modularity and layered evaluation

The proposed framework is highly modular, consisting of separate modules for performance classification, domain data processing, rules implementation, and recommendation presentation, with a simple interface between them. This modular structure enables easier development and facilitates iterative improvements of individual components.

An inherent aspect of this proposal is the layered evaluation made possible by the modular structure. Each module can be evaluated independently. Together with the focus on finding weak spots (instead of reaching optimal performance), this allows efficient development of recommendations.

7.5 Research agenda

The proposed framework is based on existing research and is directly practically applicable. It also raises many issues which would benefit from further research.

One clear direction for further exploration is the use of student modeling techniques in recommendations. Although there exists extensive research on tracing student knowledge (Pelánek 2017), we have chosen to use just abstracted performance classification. One reason is that the current research on student modeling does not properly address the issue of combining performance data from exercises with various types of interaction (e.g., written answers without any time limit vs. multiple-choice questions with time pressure). More generally, it is unclear when knowledge tracing brings sufficiently significant benefits to learning activity recommendations.

The rule-based approach to recommendations can benefit from further research in several directions. One interesting direction is identifying "missing rules" based on frequent patterns of usage that are not covered by existing rules, e.g., we are currently exploring the use of association rules over homework assigned by teachers. As a specifically important direction, we consider the clarification of processes for setting parameters and priorities of rules for strengthening knowledge (spaced repetition, interleaved practice). This is one of the key potential advantages of computerized

adaptive practice (Carpenter 2014), and even minor changes in rules and their priorities can significant impact student practice. However, due to the long-term impact of these rules, their tuning is particularly challenging.

One of the key goals of the presented framework is facilitating iterative improvement based on continuous evaluation. As we have discussed, evaluation of this type of recommender system is very challenging. It would be useful to clarify the advantages and disadvantages of various evaluation methods and provide guidance for their usage. In our discussion of the case study, we have described several caveats of basic methods for evaluation, e.g., the influence of context on click-through rates or the occurrence of Simpson's paradox. Similar issues and caveats have already been explored in other settings, e.g., in other types of recommender systems or information retrieval (Becker et al. 2007). The educational setting, however, brings multiple specific requirements, and existing techniques typically cannot be applied directly.

Funding Open access publishing supported by the National Technical Library in Prague.

References

Adomavicius, G., Tuzhilin, A.: Context-aware recommender systems. In: Recommender Systems Handbook, pp. 217–253. Springer, Berlin (2010)

Aleven, V., McLaughlin, E.A., Glenn, R.A., Koedinger, K.R.: Handbook of Research on Learning and Instruction, Chapter Instruction Based on Adaptive Learning Technologies. Routledge, London (2016)

Arroyo, I., Woolf, B.P., Burelson, W., Muldner, K., Rai, D., Tai, M.: A multimedia adaptive tutoring system for mathematics that addresses cognition, metacognition and affect. Int. J. Artif. Intell. Educ. **24**(4), 387–426 (2014)

Asadi, S., Jafari, S., Shokrollahi, Z.: Developing a course recommender by combining clustering and fuzzy association rules. J. AI Data Min. **7**(2), 249–262 (2019)

Baker, R.S.: Stupid tutoring systems, intelligent humans. Int. J. Artif. Intell. Educ. **26**(2), 600–614 (2016)

Barria-Pineda, J., Akhuseyinoglu, K., Želem-Ćelap, S., Brusilovsky, P., Milicevic, A.K., Ivanovic, M.: Explainable recommendations in a personalized programming practice system. In: Artificial Intelligence in Education: 22nd International Conference, AIED 2021, Utrecht, The Netherlands, June 14–18, (2021), Proceedings, Part I, pp. 64–76. Springer, Berlin (2021)

Becker, H., Meek, C., Chickering, D.M.: Modeling contextual factors of click rates. In: AAAI, vol. 7, pp. 1310–1315 (2007)

Bodily, R., Verbert, K.: Review of research on student-facing learning analytics dashboards and educational recommender systems. IEEE Trans. Learn. Technol. **10**(4), 405–418 (2017)

Brusilovsky, P., Vassileva, J.: Course sequencing techniques for large-scale web-based education. Int. J. Continu. Eng. Educ. Life Long Learn. **13**(1–2), 75–94 (2003)

Brusilovsky, P., Karagiannidis, C., Sampson, D.: Layered evaluation of adaptive learning systems. Int. J. Continu. Eng. Edu. Life Long Learn. **14**(4–5), 402–421 (2004)

Carpenter, S.K.: Spacing and interleaving of study and practice. In: Benassi, V.A., Overson, C.E., Hakala, C.M. (Eds.) Applying the Science of Learning in Education: Infusing Psychological Science into the Curriculum, pp. 131–141 (2014)

Castells, P., Hurley, N., Vargas, S.: Novelty and diversity in recommender systems. In: Recommender Systems Handbook, pp. 603–646. Springer, Berlin (2021)
Chen, Y., González-Brenes, J.P., Tian, J.: Joint discovery of skill prerequisite graphs and student models. Int. Educ. Data Min. Soc. **6**, 66 (2016)
Cordova, D.I., Lepper, M.R.: Intrinsic motivation and the process of learning: beneficial effects of contextualization, personalization, and choice. J. Educ. Psychol. **88**(4), 715 (1996)
da Silva, F.L., Slodkowski, B.K., da Silva, K.K.A., Cazella, S.C.: A systematic literature review on educational recommender systems for teaching and learning: research trends, limitations and opportunities. Educ. Inf. Technol. **28**(3), 3289–3328 (2023)
DeVon, H.A., Block, M.E., Moyle-Wright, P., Ernst, D.M., Hayden, S.J., Lazzara, D.J., Savoy, S.M., Kostas-Polston, E.: A psychometric toolbox for testing validity and reliability. J. Nurs. Scholarsh. **39**(2), 155–164 (2007)
Dunlosky, J., Rawson, K.A., Marsh, E.J., Nathan, M.J., Willingham, D.T.: Improving students' learning with effective learning techniques: promising directions from cognitive and educational psychology. Psychol. Sci. Public Interest **14**(1), 4–58 (2013)
Effenberger, T., Pelánek, R.: Validity and reliability of student models for problem-solving activities. In: Proceedings of Learning Analytics & Knowledge, pp. 1–11 (2021)
George, G., Lal, A.M.: Review of ontology-based recommender systems in e-learning. Comput. Educ. **142**, 103642 (2019)
Gogo, K.O., Nderu, L., Mwangi, R.W.: Fuzzy logic based context aware recommender for smart e-learning content delivery. In: 2018 5th International Conference on Soft Computing & Machine Intelligence (ISCMI), pp. 114–118. IEEE (2018)
Greer, J., Mark, M.: Evaluation methods for intelligent tutoring systems revisited. Int. J. Artif. Intell. Educ. **26**(1), 387–392 (2016)
Gulzar, Z., Leema, A.A., Deepak, G.: Pcrs: personalized course recommender system based on hybrid approach. Procedia Comput. Sci. **125**, 518–524 (2018)
Hsu, M.-H.: A personalized English learning recommender system for esl students. Expert Syst. Appl. **34**(1), 683–688 (2008)
Imran, H., Belghis-Zadeh, M., Chang, T.-W., Kinshuk, A., Graf, S.: Plors: a personalized learning object recommender system. Vietnam J. Comput. Sci. **3**, 3–13 (2016)
Iqbal, A., Oppermann, R., Patel, A., et al.: A classification of evaluation methods for intelligent tutoring systems. In: Software-Ergonomie'99, pp. 169–181. Springer, Berlin (1999)
Käser, T., Klingler, S., Gross, M.: When to stop? Towards universal instructional policies. In: Proceedings of Learning Analytics & Knowledge, pp. 289–298 (2016)
Kievit, R.A., Frankenhuis, W.E., Waldorp, L.J., Borsboom, D.: Simpson's paradox in psychological science: a practical guide. Front. Psychol. **4**, 513 (2013)
Koedinger, K.R., Corbett, A.T., Perfetti, C.: The knowledge-learning-instruction framework: bridging the science-practice chasm to enhance robust student learning. Cogn. Sci. **36**(5), 757–798 (2012)
Kolahkaj, M., Harounabadi, A., Nikravanshalmani, A., Chinipardaz, R.: A hybrid context-aware approach for e-tourism package recommendation based on asymmetric similarity measurement and sequential pattern mining. Electron. Commer. Res. Appl. **42**, 100978 (2020)
Lomas, D., Patel, K., Forlizzi, J.L., Koedinger, K.R.: Optimizing challenge in an educational game using large-scale design experiments. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 89–98. ACM (2013)
Lomas, J.D., Koedinger, K., Patel, N., Shodhan, S., Poonwala, N., Forlizzi, J.L.: . Is difficulty overrated? The effects of choice, novelty and suspense on intrinsic motivation in educational games. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, pp. 1028–1039. ACM (2017)
Maier, U., Klotz, C.: Personalized feedback in digital learning environments: classification framework and literature review. Comput. Educ. Artif. Intell. **3**, 100080 (2022)
Malone, T.W.: Making learning fun: a taxonomic model of intrinsic motivations for learning. Conative Affect. Process Anal. **6**, 66 (1987)
Manouselis, N., Drachsler, H., Vuorikari, R., Hummel, H., Koper, R.: Recommender systems in technology enhanced learning. In: Recommender Systems Handbook, pp. 387–415. Springer, Berlin (2011)
Manouselis, N., Drachsler, H., Verbert, K., Duval, E.: Recommender Systems for Learning. Springer, Berlin (2012)
Mark, M.A., Greer, J.E., et al.: Evaluation methodologies for intelligent tutoring systems. J. Artif. Intell. Educ. **4**, 129–129 (1993)

McNee, S.M., Kapoor, N., Konstan, J.A.: Don't look stupid: avoiding pitfalls when recommending research papers. In: Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work, pp. 171–180 (2006)
Mian, S., Goswami, M., Mostow, J.: What's most broken? Design and evaluation of a tool to guide improvement of an intelligent tutor. In: Proceedings of the Artificial Intelligence in Education, pp. 283–295 (2019)
Mohseni, M., Maher, M.L., Grace, K., Najjar, N., Abbas, F., Eltayeby, O.: Pique: recommending a personalized sequence of research papers to engage student curiosity. In: Artificial Intelligence in Education: 20th International Conference, AIED 2019, Chicago, IL, USA, June 25–29, 2019, Proceedings, Part II 20, pp. 201–205. Springer, Berlin (2019)
Nakamura, J., Csikszentmihalyi, M.: The concept of flow. In: Flow and the Foundations of Positive Psychology, pp. 239–263. Springer, Berlin (2014)
Paramythis, A., Weibelzahl, S., Masthoff, J.: Layered evaluation of interactive adaptive systems: framework and formative methods. User Model. User Adapt. Interact. **20**(5), 383–453 (2010)
Pavlik, P.I., Anderson, J.R.: Using a model to compute the optimal schedule of practice. J. Exp. Psychol. Appl. **14**(2), 101 (2008)
Pelánek, R.: Bayesian knowledge tracing, logistic models, and beyond: an overview of learner modeling techniques. User Model. User Adapt. Interact. **27**, 313–350 (2017)
Pelánek, R.: The details matter: methodological nuances in the evaluation of student models. User Model. User Adapt. Interact. **28**(3), 207–235 (2018)
Pelánek, R.: A classification framework for practice exercises in adaptive learning systems. IEEE Trans. Learn. Technol. **13**(4), 734–747 (2020)
Pelánek, R.: Analyzing and visualizing learning data: a system designer's perspective. J. Learn. Anal. **8**(2), 93–104 (2021)
Pelánek, R.: Adaptive, intelligent, and personalized: navigating the terminological maze behind educational technology. Int. J. Artif. Intell. Educ. **32**(1), 151–173 (2022)
Pelánek, R., Effenberger, T.: Beyond binary correctness: classification of students' answers in learning systems. User Model. User Adapt. Interact. **30**, 867–893 (2020)
Pelánek, R., Effenberger, T.: Improving learning environments: avoiding stupidity perspective. IEEE Trans. Learn. Technol. **15**(1), 64–77 (2022)
Pelánek, R., Řihák, J.: Analysis and design of mastery learning criteria. New Rev. Hypermedia Multimed. **24**(3), 133–159 (2018)
Pelánek, R., Papoušek, J., Řihák, J., Stanislav, V., Nižnan, J.: Elo-based learner modeling for the adaptive practice of facts. User Model. User Adapt. Interact. **27**(1), 89–118 (2017)
Quadrana, M., Cremonesi, P., Jannach, D.: Sequence-aware recommender systems. ACM Comput. Surv. CSUR **51**(4), 1–36 (2018)
Rahayu, N.W., Ferdiana, R., Kusumawardani, S.S.: A systematic review of learning path recommender systems. Educ. Inf. Technol. **66**, 1–24 (2022)
Raj, N.S., Renumol, V.: A systematic literature review on adaptive content recommenders in personalized learning environments from 2015 to 2020. J. Comput. Educ. **9**(1), 113–148 (2022)
Settles, B., Meeder, B.: A trainable spaced repetition model for language learning. In: Proceedings of Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1848–1858 (2016)
Šimko, M., Barla, M., Bieliková, M.: Alef: a framework for adaptive web-based learning 2.0. In: Key Competencies in the Knowledge Society, pp. 367–378. Springer, Berlin (2010)
Tabassum, N., Rehman, A., Hamid, M., Saleem, M., Malik, S., Alyas, T.: Intelligent nutrition diet recommender system for diabetic's patients. Intell. Autom. Soft Comput. **29**(3), 319–335 (2021)
Tarus, J.K., Niu, Z., Mustafa, G.: Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning. Artif. Intell. Rev. **50**(1), 21–48 (2018)
Tintarev, N., Masthoff, J.: Designing and evaluating explanations for recommender systems. In: Recommender Systems Handbook, pp. 479–510. Springer, Berlin (2010)
Vanlehn, K.: The behavior of tutoring systems. Int. J. Artif. Intell. Educ. **16**(3), 227–265 (2006)
VanLehn, K.: Regulative loops, step loops and task loops. Int. J. Artif. Intell. Educ. **26**, 107–112 (2016)
Vassileva, J.: Reactive instructional planning to support interacting teaching strategies. In: Proceedings of the 7th World Conference on AI and Education, pp. 334–342. AACE, Charlottesville (1995)
Walek, B., Fojtik, V.: A hybrid recommender system for recommending relevant movies using an expert system. Expert Syst. Appl. **158**, 113452 (2020)

Walkington, C.A.: Using adaptive learning technologies to personalize instruction to student interests: the impact of relevant contexts on performance and learning outcomes. J. Educ. Psychol. **105**(4), 932 (2013)
Weibelzahl, S.: Problems and pitfalls in the evaluation of adaptive systems. In: Adaptable and Adaptive Hypermedia Systems, pp. 285–299. IGI Global (2005)
Weibelzahl, S., Paramythis, A., Masthoff, J.: Evaluation of adaptive systems. In: Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization, pp. 394–395 (2020)
Zhang, Y., Chen, X., et al.: Explainable recommendation: a survey and new perspectives. Found. Trends Inf. **14**(1), 1–101 (2020)
Zinkevich, M.: Rules of machine learning: best practices for ML engineering. http://martin.zinkevich.org/rules_of_ml/rules_of_ml.pdf. Accessed 14 Aug 2023 (2017)

Radek Pelánek received a Ph.D. degree in computer science from Masaryk University, Brno, Czech Republic, for his work on formal verification. Since 2010 his research interests have focused on areas of educational data mining and learning analytics. Currently, he is the leader of the Adaptive Learning group at Masaryk University and is interested in both theoretical research in user modeling and the practical development of adaptive learning systems. Tomáš Effenberger received a Ph.D. degree in computer science from Masaryk University, Brno, Czech Republic, for his work on data-driven methods for improving online learning systems. After research internships at the University of Oxford and Google AI, he now works as a data scientist and content curator in Umíme, a leading online learning system in the Czech Republic. Petr Jarušek received a Ph.D. degree in computer science from Masaryk University, Brno, Czech Republic, for his work on student modeling in problem solving. After finishing his doctoral studies, he founded Umíme, which became a leading online learning system in the Czech Republic.