Towards Distributed Intelligent Tutoring Systems Based on User-owned Progress and Performance Data

Abstract

The use of recommendation engines to personalise students' learning experiences can be beneficial by providing them with exercises that are tailored to their knowledge. However, the use of these systems often comes at a cost. Most learning or tutoring systems require the data to be stored locally within a proprietary database, limiting the freedom of the learner as they move across different systems during their learning journey. In addition, these systems might potentially cause additional stress, as the learner might feel observed without knowing who has access to their learning progress and performance data. We propose a solution to this problem by decentralising learning progress and performance data in user-owned Solid Pods. We outline the proposed solution by describing how it might be applied to an existing environment for programming education that already includes research on how to align difficulty levels of exercises across different systems.

Keywords

Personal Data Vault, Intelligent Tutoring Systems, Exercise Recommendation

1. Introduction

The use of Intelligent Tutoring Systems (ITS) to adapt the recommendation of exercises based on individual learners has proven to be effective [1]. Evidence suggests that more personalised learning leads to increased learner agency, self-reliance and motivation [2]. However, the move towards smart learning systems is not without costs. Intelligent Tutoring Systems typically require that all of the exercises are stored in one centralised system that is often also monitored by teachers and persons who are responsible for ultimately grading the students. Previous studies revealed that implementing electronic performance monitoring can result in lower job satisfaction, increased stress levels, reduced autonomy, and it can further be perceived as a violation of trust [3]. In ideal circumstances, the learning environments should also offer a safe space for students to practise on problems they find interesting or challenging, without having to worry about how they will be perceived by teachers. To give an example, a student might be worried that the perception of the teacher might change if they are still shown to be struggling with material from previous years.

In the following, we propose a solution to this problem based on the Solid specification where (a) *the same user progress can be used across multiple learning applications* to provide a smooth flow state and (b) *a student can opt in to share progress with individual applications for training the model*, but without having to provide teachers access to their entire performance for individual exercises. We discuss the solution by answering some open questions posed by Malaise and Signer [4]. They proposed the *Explorotron* prototype for programming education. In this prototype, there are multiple smaller sub-applications called *study lenses*, each taking

The 1st Solid Symposium Poster Session, co-located with the 2nd Solid Symposium, May 02 – 03, 2024, Leuven, Belgium

a source code file and generating exercises based on certain difficulty levels matching the PRIMM methodology [5]. While their tool offers the possibility to suggest learning experiences based on logical progression through built-in study lenses, the authors conclude with a few major challenges to overcome:

- Students can generate exercises from any source file, so generated content is never used by other students, leading to a cold start problem that makes recommendations challenging.
- Students should be free to decide which data they want to share and with whom they want to share it.
- Third-party developers should be able to contribute custom exercise generators (study lenses) transparently—the same recommendations/profiling should work across the board.

2. Solution

For our demonstration, we assume that student modelling happens through the use of Bayesian Knowledge Tracing (BKT) [6] which is a common approach in the design of Intelligent Tutoring Systems. However, note that with some modifications, other approaches could also be supported. BKT is a probabilistic model that requires four parameters to be fitted for each learning object as illustrated in Figure 1.

- $P(L_t)$: The probability that the topic being covered by the learning object is mastered at time t. A value $P(L_0)$ has to be provided to indicate the chance the user knows the topic before attempting any exercise. Note that the topics are defined as a combination of the knowledge topic and the difficulty level as defined later in Section 2.1.
- $P(S)$: The probability that the user makes a *slip*, i.e. gets it wrong even though they know the topic.
- $P(G)$: The probability that the user makes a lucky guess and gets the answer right even though they do not know the topic.
- $P(T)$: The probability that the user actually learns the topic while performing this exercise.

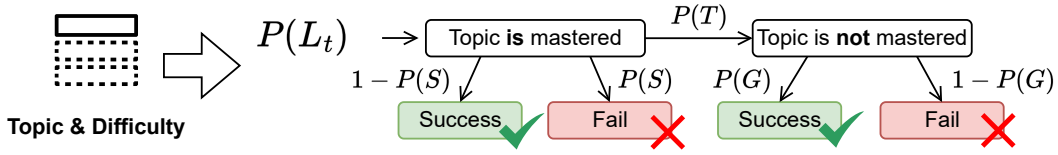

Figure 1: Example of the Bayesian Knowledge Tracing model per topic and difficulty pair

Based on these values, we can then calculate the updated probability that a topic (e.g. "Arrays") is mastered each time the user is presented with a new exercise about this topic (L_{t+1}) based on whether they succeeded or failed solving the exercise. It is important to note that all these probabilities are exercise dependent and will as such not be shared across the applications except for $P(L_t)$. This value is the probability that the user knows a topic, which can be shared across any application as long as they are referring to the same topic.

We aim to store the result of every exercise the user performs within a personal data vault, together with the calculated $P(L_t)$ for every topic and difficulty level pair as shown in Figure 2. Developers of educational applications could then request limited-time access to the entries of all of their users in order to fit their models, without the need to permanently own all the data. Shared information, such as the $P(L_t)$, will further make it easier for other developers to contribute their own extensions that provide exercise recommendations.

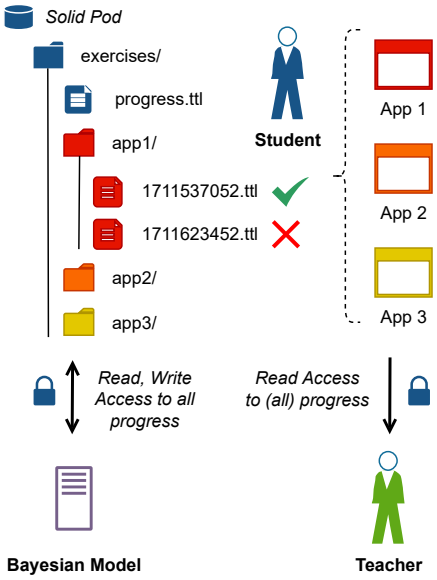

Figure 2: Overview of individual apps storing exercise results in a Solid Pod

```
<> a schema:ReplyAction; # A reply to a question
  dct:created "2024-03-28T11:57"^^xsd:dateTime;
  schema:agent <http://.../profile/card#me>;
  # Exercise question
  schema:parentItem [ a schema:Question;
    schema:name "What is the output of ...?"@en;
    schema:educationalLevel primm:Predict;
    schema:eduQuestionType "Flashcard"@en;
    foaf:topic dbr:Array;
    foaf:agent <https://app1.org/>. # Creator
  ];
  schema:result [ a schema:Answer; # User answer
    schema:answerExplanation [ ... ];
    schema:review [ a schema:Review;
      # The grade of the answer (e.g. 5 out of 5)
      schema:reviewRating 5; schema:bestRating 5.
    ]
  ].
```

Listing 1: Example reply on a question generated by app #1 using prefixes `dbr` [7], `schema` [8], `foaf` [9] and `dct` [10]

Users can already indicate their topics of interest in their Solid Pod using the topics of interest predicate from the `foaf` vocabulary [9], which can aid educational recommendation systems to provide exercises that interest the user. Together with data about the past performance, we can create personalised learning paths for users based on their strengths and weaknesses from multiple sources without creating any vendor lock-in and without claiming ownership of the students' data.

In addition, the progress of these exercises and topics can optionally be shared with an educator or mentor as illustrated in Figure 2, allowing for personalised feedback and guidance to further support a user's learning journey. The fine-grained control over the data also allows students to share the information on topics they are currently covering in class, while preventing the need to share that they are still working on other topics that they do not want to share.

2.1. Exercises and Progress

Each topic has a set of exercises with their individual difficulty level. With our proposed solution we use the existing PRIMM principles [5] to identify the five stages of difficulties for exercises. These stages include the ability to *predict* the output of a piece of code, *run* a piece of code,

investigate a piece of code and answer some questions about its structure, *modify* a piece of code so the behaviour changes to a new desired result and finally, *make* a similar piece of code from scratch. Together, these five principles identify the logical progression of difficulty levels with which to look at a piece of code from the perspective of a student. Listing 1 illustrates an example exercise and the answer given to this exercise that includes the difficulty level[1] using the `educationalLevel` predicate.

A further extension of the architecture would allow third-party developers to share the capabilities of their plugins with regards to what type of content their plugins could provide so the recommendation engine can be a fully separate component querying all content providers included by the user without it being centrally controlled by one individual.

```
<#progress_arrays_predict> a sosa:ObservableProperty ;
    rdfs:label "Prediction progress"@en ; ssn:isPropertyOf <#me> ;
    foaf:topic dbr:Array ; schema:educationalLevel primm:Predict .
<#1711623452> a sosa:Observation ; # P(L_(t+1))
    sosa:usedProcedure dbr:Bayesian_Knowledge_Tracing ;
    dct:created "2024-03-28T11:57"^^xsd:dateTime ;
    foaf:agent <https://app1.org/> ; sosa:observedProperty <#progress_array> ;
    sosa:hasResult [ qudt:floatPercentage "38.12"^^xsd:float ] .
<#1711537052> a sosa:Observation ; # P(L_t)
    sosa:usedProcedure dbr:Bayesian_Knowledge_Tracing ;
    dcterms:created "2024-03-27T11:57"^^xsd:dateTime ;
    foaf:agent <https://app1.org/> ; sosa:observedProperty <#progress_array> ;
    sosa:hasResult [ qudt:floatPercentage "25.0"^^xsd:float ] .
```

Listing 2: Multiple observations of user progress for a particular topic and difficulty level

Listing 2 shows multiple observations of user progress for a particular topic at a given time (i.e. $P(L_t)$). Each observation, described using the SOSA ontology [11, 12], is created by an application. This progress is shared, allowing each educational application to use these observations to determine the current knowledge on a topic.

3. Conclusions

In this paper, we analysed a use case of a digital learning platform that helps students to learn programming but currently has some open challenges on how personalisation and interoperability with third-party developers might be achieved without taking away the learner's ownership of their data. We then proposed a novel solution to support those tasks by allowing users to store all their exercise results in Solid Pods and giving them full control on how and when applications can access this data. In this work, we assumed that both the users and the application developers could be fully trusted. If more validation and authority is needed, additional layers can be built on top of this system, for example by using a blockchain to verify claims, as described in [13, 14].

[1]The `primm:` vocabulary represents a trivial vocabulary to describe the difficulty levels in programming exercises

References

[1] J. A. Kulik, J. D. Fletcher, Effectiveness of Intelligent Tutoring Systems: a Meta-Analytic Review, Review of educational research 86 (2016).

[2] V. Prain, P. Cox, C. Deed, J. Dorman, D. Edwards, C. Farrelly, M. Keeffe, V. Lovejoy, L. Mow, P. Sellings, et al., Personalised Learning: Lessons to be Learnt, British Educational Research Journal (2013). doi:10.1080/01411926.2012.669747.

[3] R. Siegel, C. J. König, V. Lazar, The Impact of Electronic Monitoring on Employees' Job Satisfaction, Stress, Performance, and Counterproductive Work Behavior: A Meta-Analysis, Computers in Human Behavior Reports 8 (2022). doi:10.1016/j.chbr.2022.100227.

[4] Y. Malaise, B. Signer, Explorotron: An IDE Extension for Guided and Independent Code Exploration and Learning (Discussion Paper), in: Proceedings of the 23rd Koli Calling International Conference on Computing Education Research, Koli Calling '23, 2024. doi:10.1145/3631802.3631816.

[5] S. Sentance, J. Waite, M. Kallia, Teaching Computer Programming With PRIMM: A Sociocultural Perspective, Computer Science Education 29 (2019). doi:10.1080/08993408.2019.1608781.

[6] O. Bulut, J. Shin, S. N. Yildirim-Erbasli, G. Gorgun, Z. A. Pardos, An Introduction to Bayesian Knowledge Tracing with PyBKT, Psych 5 (2023).

[7] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, Dbpedia: A nucleus for a web of open data, in: K. Aberer, K.-S. Choi, N. Noy, D. Allemang, K.-I. Lee, L. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber, P. Cudré-Mauroux (Eds.), The Semantic Web, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 722–735.

[8] A. Iliadis, A. Acker, W. Stevens, S. B. Kavakli, One Schema to Rule them All: How Schema.org Models the World of Search, Journal of the Association for Information Science and Technology (2023). doi:https://doi.org/10.1002/asi.24744.

[9] D. Brickley, L. Miller, FOAF Vocabulary Specification 0.91, 2007.

[10] S. Weibel, J. Kunze, C. Lagoze, M. Wolf, Dublin Core Metadata for Resource Discovery, Technical Report, 1998.

[11] K. Janowicz, M. Compton, The Stimulus-Sensor-Observation Ontology Design Pattern and its Integration Into the Semantic Sensor Network Ontology, in: Proceedings of SSN 2010, International Workshop on Semantic Sensor Networks, 2010.

[12] K. Janowicz, A. Haller, S. J. Cox, D. Le Phuoc, M. Lefrançois, SOSA: A Lightweight Ontology for Sensors, Observations, Samples, and Actuators, Journal of Web Semantics 56 (2019). doi:10.1016/j.websem.2018.06.003.

[13] N. Chowdhury, M. Ramachandran, A. Third, A. Mikroyannidis, M. Bachler, J. Domingue, Towards a Blockchain-based Decentralised Educational Landscape, in: Proceedings of the Twelfth International Conference on Mobile, Hybrid, and On-line Learning, 2020.

[14] A. Mikroyannidis, Blockchain applications in education: a case study in lifelong learning (2020).