Augmenting Digital Textbooks with Reusable Smart Learning Content: Solutions and Challenges*

Jordan Barria-Pineda[1 [0000−0002−4961−4818]], Arun Balajiee Lekshmi Narayanan[1 [0000−0002−7735−5008]], and Peter Brusilovsky[1 [0000−0002−1902−1464]]

University of Pittsburgh, Pittsburgh PA 15260, USA

Abstract. A powerful set of educational tools has emerged over the last decade with the rise in the adoption of online adaptive learning content. An increasingly popular tool in this space is the "intelligent textbook" as a platform to support and distribute content for e-learning, given its resemblance with real-life physical books. Existing efforts in this direction include the development of digital textbooks where both textual content and interactive learning activities (i.e., examples, problems, etc.) are carefully handcrafted by the authors so that they are perfectly placed to follow the knowledge acquisition-practice flow. However, this approach is very time-consuming, and it requires the work of high-expertise authors. In this work, we suggest and discuss a scalable solution: we take existing digital textbooks and augment them by using repositories of existing online learning material associated with the subject matter. We present our current work in this direction and discuss challenges and opportunities for the future work.

Keywords: Smart Learning Content · Intelligent Textbooks · Educational Recommendations

1 Introduction

A gradual switch from paper-based to "electronic" textbooks (e-textbooks) opened an exciting opportunity to extend these classic learning tools with functionalities not previously available in paper format. Among the most appealing and popular ways to extend textbooks with new functionalities is converting examples and problems, a traditional component of textbooks in many domains, into interactive learning activities. This approach makes textbooks truly interactive and augments learning by reading with learning by doing.

One of the first domains to embrace this kind of interactive textbook was computer science education (CSE) where the development of interactive learning activities from algorithm animations to automatically-assessed programming problems was a popular research direction. The need to integrate interactive learning

activities with online textbooks has been extensively discussed by the computer science education community for many years [24] and some best examples of interactive textbooks have been produced for computer science subjects. Among these examples are ELM-ART [7], the first adaptive textbook with interactive problems and examples for learning LISP, OpenDSA [13], the first open-source infrastructure for collaborative construction of e-textbooks with interactive animations and problems (originally developed for Data Structures and Algorithms course), and RuneStone books [12] a popular infrastructure for presenting online textbooks for programming augmented with interactive learning activities. These and other interactive textbooks have been extensively evaluated in various learning contexts and their effectiveness was convincingly demonstrated [31, 11, 21].

However, the current platforms for the development and delivery of interactive textbooks for CSE share the same problem: the "custom" nature of their production. These textbooks are expected to be developed "as a whole" for a specific purpose, with text and interactive problems developed and integrated together as a part of the authoring process. This approach allows developing excellent examples of interactive textbooks but doesn't support scaling up this process. For each "holistically developed" interactive textbook, there are dozens of professionally authored textbooks on the same subject that are not augmented with interactive content because this option has not been considered at the time of their creation. At the same, there are large repositories of interactive learning content of different types that could be used to augment these books. A missing piece in the infrastructure is the integration of an arbitrary textbook with its corresponding interactive content.

An important step towards building this infrastructure was done in OpenDSA project [13], which offered an opportunity to connect any LTI-compatible interactive content to OpenDSA textbooks. However, it is still focused on custom-built textbooks and doesn't support existing textbook. The project presented in this paper attempts to take the next step in this direction and make both textbooks and integrative learning content reusable. Our goal is to build an infrastructure that allows turning any textbook available in electronic format (such as PDF) into an interactive textbook by augmenting it with interactive learning content from existing repositories. This paper presents an important component of this infrastructure - and interface that support augmentation of existing books with interactive content without breaking the structure of these textbooks. In the following sections, we present our current implementation of this interface, demonstrate the approach for integrating smart content into textbook structure, and discuss future work in this direction.

2 Related Work

Multiple research efforts have been carried out during the last decade in order to develop technology-enhanced textbooks. Electronic Textbooks (e-textbooks) support content distribution at scale in different formats and for different pur-

poses. In recent years, there have been many discussions that project what technological enhancements could surround the use of intelligent e-textbooks in education [23]. Among these discussions, a few noteworthy contributions such as the use of intelligent question-asking [19], intelligent tutoring [29], and augmentation of assessment questions [10] bring to the light the possibilities of interesting enhancements in smart digital textbooks. On the one hand, these technological enhancements could be implemented as artificially intelligent agents or systems in e-textbooks that deliver, recommend or scaffold the learner's needs while reading [32]. On the other hand, it could be possible to integrate reusable smart content that are adaptive to the needs of the user, without necessarily adapting or personalizing the system behaviour to the learner. In general, the main idea has been to maintain the affordances of physical textbooks combined with the capabilities of web pages. However, some efforts have been made to incorporate the design of the novel functionalities for students that could expand the potentialities of intelligent textbooks [30].

In our work, we take this second route of augmentation of intelligent textbooks with adaptive, personalized learning material presented as Smart Learning Content (SLC) [5]. Typically, there are 5 different levels of SLCs [5], namely,

1. **Level 1** SLCs are independent of the delivery platform for learning. In this setup, the variables used by the SLC to personalize content do not persist, once the session is closed and is stateless.
2. **Level 2** The SLC and the delivery platform for learning are integrated into a single system and the delivery platform saves the data produced by the SLC, which is used across several user sessions. A limitation of this setup is that the SLC would need to be developed specifically for the delivery platform and cannot exist outside of the system. For example, CodeAcademy [1], KhanAcademy [2], Brilliant [3]
3. **Level 3** All content in the SLCs is internal to the platform, but the delivery platform supports multiple SLCs. For example, OpenDSA [13]
4. **Level 4** The platform supports multiple SLCs and allows the use of external content, using proprietary protocols to retrieve the external content. For example, BlueJ or Moodle with plug-in support, TestMyCode [28], A+ [18] and JavaGuide [15].
5. **Level 5** The platform supports multiple SLCs that use standard protocols, such as Learning Tools Interoperability (LTI), allowing for maximum flexibility. For example, LTI with Moodle [4].

Among these levels, our earlier discussions cover implementations that could be considered as level 2 SLCs [8]. We also discussed implementations that utilize the benefits of Learning Tools Interoperability (LTI) [2] to integrate early implementations of level 5 SLCs in the intelligent textbooks. In this work, we explore

[1] http://codeacademy.com
[2] https://khanacademy.org
[3] https://brilliant.org
[4] https://docs.moodle.org/400/en/LTI_and_Moodle

possibilities for an implementation that could meet the gold standards discussed as levels 3 and 4 in our prior work, that is, to support multiple SLCs that are both native and external resource recommendations to the delivery platform. In this case, we set the platform content delivery to be the intelligent textbook.

3 Reading System to Support Social Aspects of Learning

For our implementation in this work, we utilize our reading system [3], a platform for social navigation and social aspects of learning. This system offers many features for the students in a course such as textual annotations and social comparison plots (for comparing self-progress with the progress of the rest of the class peers). The system is intended to support and enhance self-regulated learning in the student using this to read e-textbooks. The system distributes course content by leveraging the use of open and accessible textbook materials on topics in various courses ranging from Introductory Information Retrieval to Introductory Object-Oriented Programming. It provides helpful visualizations to view the reading progress over the duration of the course, with interactive multiple-choice quizzes at the end of each section, presented within a pop-up window. Considering these aspects, this system provides an excellent platform to support extensible SLCs like animated programming exercises [25], programming construction examples [14] and external content recommendations (e.g., Wikipedia articles) [22]. By default, there are native content recommendations available in an intelligent textbooks system, which could be considered to satisfy the principles of level 3 SLCs.

4 Augmenting Reusable Smart Learning Content

We set the goals for level 3 and 4 SLCs, which we describe again here,

1. **Level 3** Integrate Multiple SLCs that are native to the platform such as OpenDSA [13]
2. **Level 4** Support multiple SLCs on the platform such that external content can be integrated into the system.

Currently, we implemented two ways to integrate SLC into a textbook: a list of recommended videos (see the red tab in Fig. 1) and a list of statically attached interactive exercises (see the green tab in Fig. 1). The video interface was developed for an information retrieval textbook. It shows recommended videos using thumbnails, which work as links to related content on YouTube[5] and multimedia sharing websites. These implementations could consider a ranking and rating-based approach to listing the content to allow the factor of "human-in-the-loop" recommendations to support and enhance intelligent recommendations to the users of these systems and their students in these courses.

[5] https://www.youtube.com

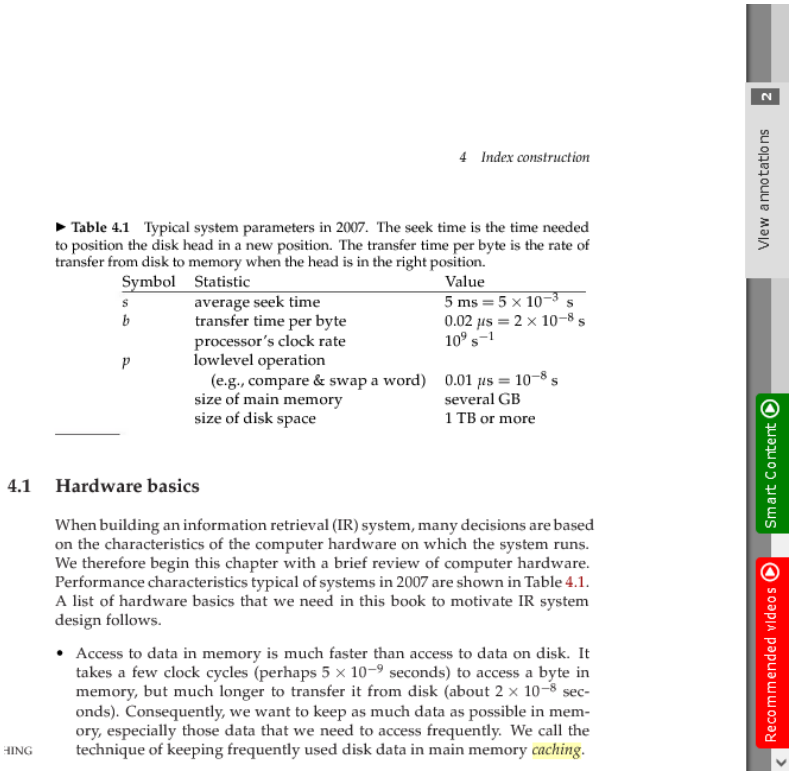

Fig. 1. Smart Content (green tab) along with the Video Recommendations (red tab) in an information retrieval textbook

The second way (a list of interactive programming exercises) was developed for e-textbooks on introductory programming. The list of available exercise types with descriptions is provided in Table 1. These exercises range from simple problems that test the student's understanding of the inputs or outputs of a given program to puzzles that can be solved in several steps. Interactive and animated examples could make the process of reading and understanding the code more engaging for the reader. This could scaffold a student's learning in their process of understanding a course on introductory concepts in programming. Such an SLC integration could possibly turn the mundane process of reading a textbook into a rich, interactive experience that offers possibilities for hands-on content experimentation. Further, students who are curious learners can explore the concepts discussed on a page with a related live, interactive examples to keep them engaged.

Table 1. Types of Systems for Integrated Python Programming Exercises

System	Types of Exercises
QuizPET [6]	Parameterized code tracing problems for Python with automatic assessment
PCEX [14]	Program construction examples in an engaging, interactive form in order to increase motivation
WebEX [4]	Web-based programming examples in Python & other languages
2D Parsons [17]	A 2D version of Parson's puzzles for Python
PCRS [33]	Programming problems that provide incomplete skeleton code and checks the answers using a set of tests
Jsvee [25]	Animated programming examples to visualize the program steps

5 Proof-of-Concept Implementations

We collected a set of SLCs from various sources, in order, to offer a wide range of online learning activities to students. The available content ranges from a low level of interactivity (i.e., educational videos and worked-out examples) to more interactive activities (i.e., parsons problems and coding-from-scratch problems). In this paper, we will focus on two courses as study cases:

1. A Graduate course on Information Retrieval based on an open source textbook
2. An Undergraduate course on Programming in Python with the main textbook for reference, "Python for Everybody"[6].

5.1 An Information Retrieval Textbook with Smart Content

For the Information Retrieval course we focused on augmenting the existing reading resources within the system with educational videos scraped from the web, more specifically from the YouTube platform[7]. We decided to present this augmentation as non-intrusive recommendations shown at the right margin of the active page. The steps we followed for generating these educational video recommendations are presented below:

1. *Candidate videos' collection*: Collecting a set of videos from YouTube covering the core Information Retrieval concepts presented in the course. We determined this set of concepts by automatically extracting them from the textual content of the course reading sections by following one of the approaches

[6] https://www.py4e.com
[7] https://www.youtube.com/results?search_query=information+retrieval

for presented by Thaker et.al. in [27]. In this concept extraction approach, first noun phrase chunking is performed to obtain candidate keyphrases from a text and then the keyphrases are ranked based on their tf-idf score [8]. As an outcome of this process, each section of the course ends up associated with a list of concepts, which are covered in the corresponding text. With this list of concepts (mainly unigrams and bigrams) we prepared a set of queries in the format *"information retrieval concept"* (e.g. *"information retrieval zipf law"*). We executed the queries by using the YouTube Search API, and for each query that is executed, we considered only the top 20 videos as potential candidates.

2. *Videos' textual representation*: We proceeded to get the transcripts of the candidate videos by using the *youtube-dl* API[9], so we could get the content presented in a textual format. Additionally, we concatenated the title and the description of the video (which was useful for having a representation for videos without an available transcript).
3. *Videos' concept extraction and initial relevancy filter*: Next step was extracting the keyphrases from the textual representation of the video, which we consider as a proxy for identifying the concepts covered there. For that task we simply looked for exact textual match between the textual transcripts and the list of concepts of the corresponding section. Thus, for each video we ended up with a list of concepts (keyphrases) representing the covered content. As an initial filter of video relevancy, we calculated the proportion of concepts covered by the video and the set of concepts of each section. If the match of the concepts presented in the video and the section was over 5%, we considered the video as "initially relevant".
4. *Reading-Video Similarity Calculation*: We computed the similarity between the initially relevant videos and the reading section by using different text similarity metrics like Cosine and Jaccard similarity and applying tf-idf as pre-processing of the textual data.
5. *Video Recommendations' Presentation*: Finally, we sort the videos descending by the similarity score calculated in the previous step and show only the top 20 on the tab (see Fig. 2). We display the video title along with its thumbnail, which, when clicked, plays a video within a pop-up window (see Fig. 3)

As a result, while navigating the reading sections through the online reader students have also the option of deepening their acquired knowledge through the recommended videos list. These external materials present the same or part content from a different perspective or at a different level of detail. Finally, in order to get feedback from the students about the suitability of the recommended material for the section that they are reading, we added a section within the video watching for them to express their opinion about the quality of the recommendation (see the top right section in Fig. 3).

[8] https://github.com/khushsi/ConceptExtractor
[9] https://youtube-dl.org/

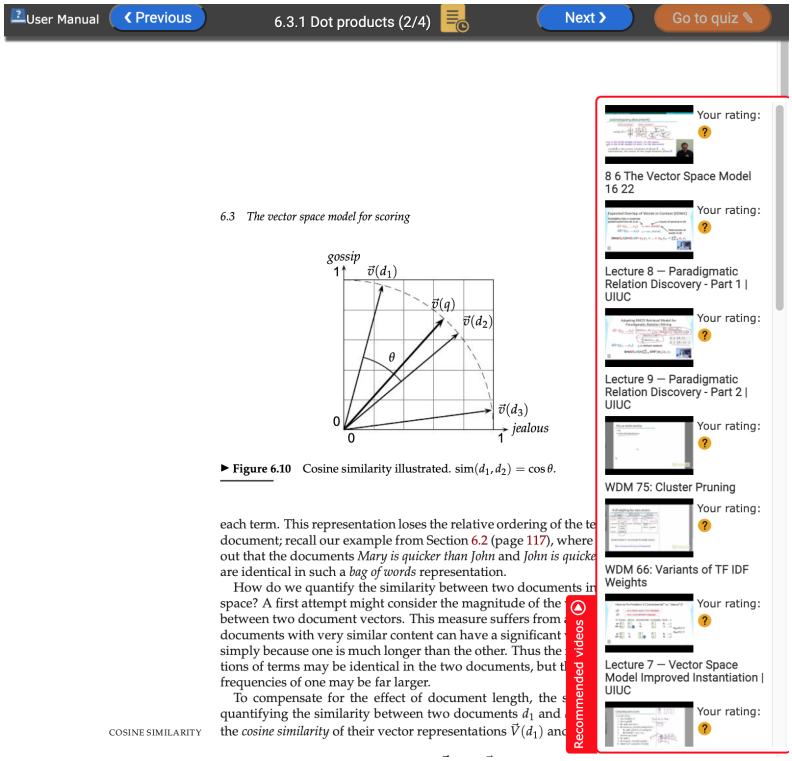

Fig. 2. Video Recommendations Interface. In this figure, we show a list of videos with their thumbnails to an Information Retrieval Textbook

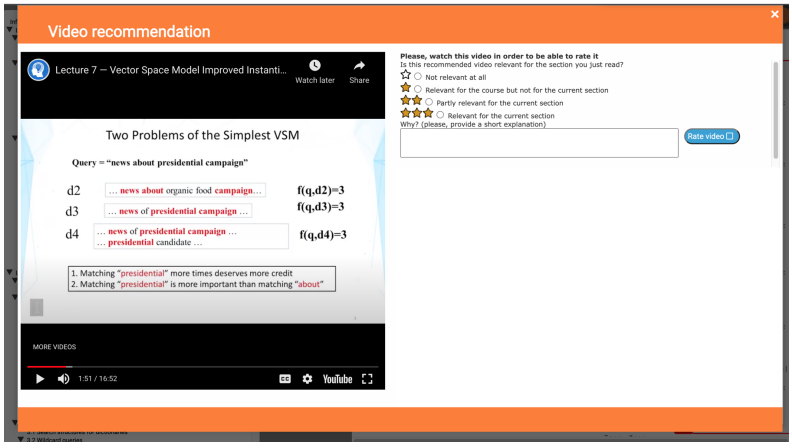

Fig. 3. Video Recommendations Presentation and Rating Interface. This modal is displayed when a particular video is selected from the video recommendations interface, allowing for "human-in-the-loop" interaction with the system, such that the recommended videos are updated by the students' rating to these videos

5.2 A Python Programming Textbook with Smart Content

To support a Python Programming course we augmented a popular textbook "Python for Everybody" with smart content. This textbook is available in several formats, including as a PDF[10]. It consists of several chapters that cover the basics of Python programming. The textbook starts with delivering the course content from scratch, going into sequential topics, and keeping the target audience as novice or beginner-level programmers in Python. We use this setup to experiment with SLC implementations to help practise programming in Python with a set of worked out examples and programming problems. We target SLCs that cannot be directly covered within the text. We think that the programming exercises could be presented as a list of short problems related to the material being read in a given page, section or chapter of the book.

To test our current infrastructure, we attached a range of smart learning content for Python to various sections of the textbook as shown in Fig. 4. When a link to an SLC item from the list of entries for smart content is clicked, it launches a dialog instance with the specific programming example or problem. For example, Fig. 5 shows a code tracing problem from QuizPET system (Quizzes for Python Educational Testing) [6]. Another kind of programming exercises for Python that we made available in the book are interactive worked examples of program construction from PCEX system with step-wise program explanations and walk-through (Fig. 6. These exercises allow for the student to focus on specifics of a given program).

In total, we demonstrated the ability to connect six types of SLC worked examples and problems listed in Table 1. A more detailed description of these SLC types could be found in [14]. When augmenting the book with SLC, we considered Python programming activities that are related to the topics covered in the text of a particular page of section in a chapter. These programming activities use the knowledge or concepts covered in the textbook up to that point and avoid the concepts that will be covered later in the textbook.

In the future version, we hope to provide a smart textbook authoring system for course instructors, which will allow them to augment the same textbook with SLC that they want to use in their classes. A prototype of this authoring system with learning analytics support can be found in [1]. We also plan to support the authoring process with instructor-focused content recommender system [9]. This could be considered as our long term goal for smart content, but in the current work of smart content design for programming exercises, we only focus on the interface for delivering SLC to students through a textbook.

6 Discussion and Future Work

With our implementation, we show that it is possible to integrate and augment e-textbooks with multiple SLCs. They are available as non-intrusive sidebars for the reader to explore material relevant, without losing focus on the main

[10] https://www.py4e.com/book

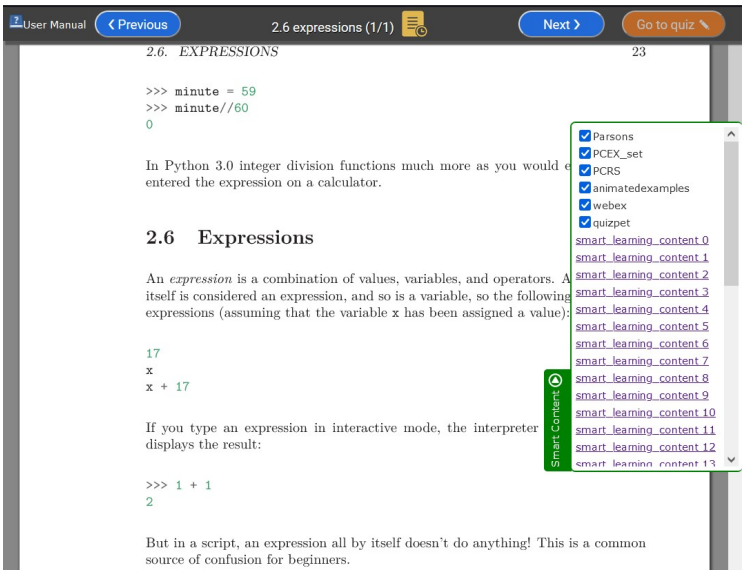

Fig. 4. List of smart exercises displayed to practice Python programming. These are links to the different programming exercises hosted on external repositories, to support Level 4 smart learning content

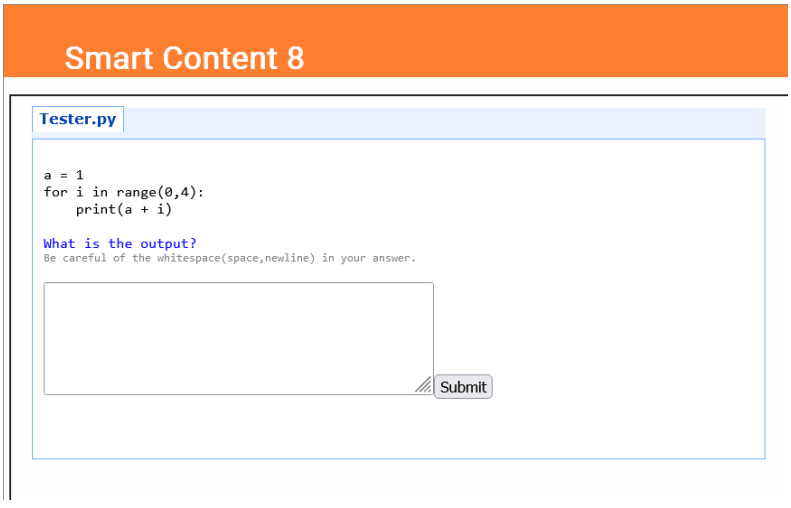

Fig. 5. Smart Content Modal with a Programming Exercise in Python. In this example, the student has to type the correct output to the program in the textbox below for system to evaluate

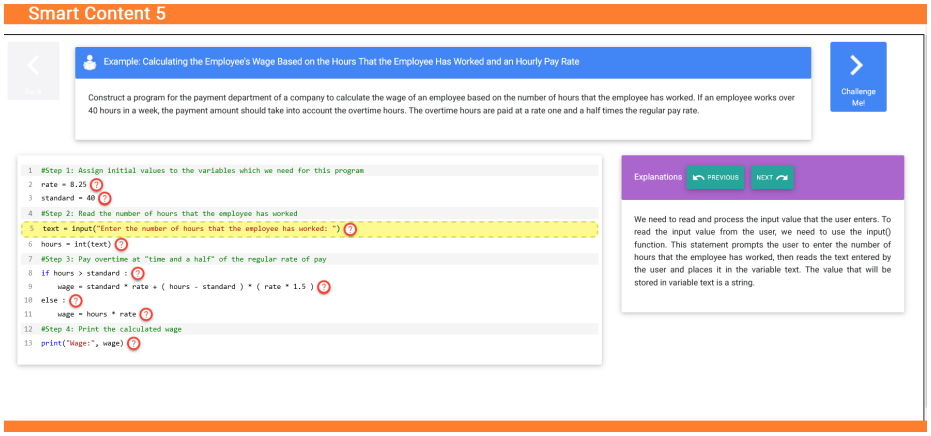

Fig. 6. Smart Content Modal with a programming exercise in Python. In this figure, the student is asked to explain the different lines in a given program, broked down into smaller steps and compare them with the standard explanations that the system expects to be appropriate at those lines

text content of the intelligent textbook. Dynamic SLC recommendations provided at different levels of granularity by relevance (given section, chapter, page or paragraph) is our next step to explore. Determining the most appropriate granularity and difficulty level of the programming exercises in the smart content to act as useful recommendations that scaffold students' learning. Further, adaptations could model the patterns of user or student interactions with the system to fine-tune the recommended SLC, further governed by the reader's control on the ("human-in-the-loop") curation of the SLCs listed with rating and ranking features. SLC of other types could take no inputs from the user, but present as passive recommendations in relation to the concepts covered at a page or section. Learner-sourced approaches to recommend questions [20, 16] is another interesting research opportunity to explore and address the challenge of dynamic content allocation. Questions that are most relevant to a page, section or chapter could be dynamically curated in the side panel. The learner-sourced SLC could be generated by peers taking the course or students who took the course. This material could be rated and ranked by the current users to improve the recommendations provided. Reusing resources in this manner could potentially open the doors to exemplary SLCs integrated into intelligent textbooks for other learner content delivery systems. These opportunities can be realized by overcoming a few challenges discussed below.

6.1 Challenges

In our implementation towards integrating multiple SLCs, we find that allocating the right content could be a potential challenge. While it is possible to support personalized, integrable and adaptive SLCs for specific chapters or sec-

tions within intelligent textbooks on different topics, it is a challenge to make it scale up to different topics and courses in these system implementations. Another challenge is that for instructors teaching these programming courses, as discussed by Chau et. al. [9], the content allocation for all the intended concepts in the course may not be possible and this makes this implementation potentially static. Further, a scope for future exploration is the allocations that adapts to the teachers' understanding of the course topics [26] is another challenge. It would especially be interesting to include a smart content that is modeled by the topic and the knowledge levels of the course instructors. Hence, the presented content would then augment their understanding of the topics covered in the course. All these three challenges consider a "human-in-the-loop" implementation. A final challenge is to integrate an SLC to augment the content presented in the digital textbook as a recommender system that involves less human intervention to improve its personalization. An example to support such an integration would be recommendations to external web resources like Wikipedia with additional information is not native to the content available within the text, but augments the information provided without much scope for the user to rate or rank these recommendations to match their personal choices. An implementation in this direction [22] sets possible paths for us to explore as augmented SLC in our future work and as a means to overcome this challenge. In the light of understanding and overcoming these challenges, we will be able to explore and support more types of dynamic SLCs in our future iterations of intelligent textbooks.

7 Conclusions

In this work, we present an interesting perspective on integrating smart learning content (SLCs) in intelligent textbooks as the delivery platform. Along with the possibility of supporting multiple SLCs, they could be native and external resources using open, proprietary protocols (levels 3 and 4 SLCs) for retrieval. This meets our goal that we set forth of building an infrastructure that could turn any ordinary e-textbook into an intelligent, adaptive and interactive textbook. Although not our goal to begin with, since our implementation uses the resources that are not *native*, but external to the intelligent textbooks framework (we benefit from using the SLC repositories developed by others), we present a system that is flexible, suggesting that the learning content delivery platform can be interchangeable (level 5). The seamless integration of the SLCs into intelligent textbooks, allows for the possibility of interactive and engaging learning content delivery platforms for curious learners. In the long term, this allows for better adoption of enhanced intelligent e-textbooks. Finally, we discuss the challenges encountered while making scalable integration of SLCs into the deliver platform. We discuss existing solutions that could allow us to overcome these challenges. Technical advancements in the not so distant future could help address these challenges with efficient protocols for seamless augmentation of smart learning content without breaking the structure of the e-textbooks.

Acknowledgements We acknowledge the help offered by our colleagues in the implementation of parser and smart learning content allocation in our intelligent textbook implementation. Also, the work of one of the authors was funded by CONICYT PFCHA/ Doctorado Becas Chile/ 2018 - 72190680.

References

1. Albó, L., Barria-Pineda, J., Brusilovsky, P., Hernández-Leo, D.: Knowledge-based design analytics for authoring courses with smart learning content. International Journal of Artificial Intelligence in Education **32**, 4–27 (2022)
2. Barria-Pineda, J., Akhuseyinoglu, K., Brusilovsky, P.: Learning content integration into an electronic textbook for introductory programming (2019)
3. Barria-Pineda, J., Brusilovsky, P., He, D.: Reading mirror: Social navigation and social comparison for electronic textbooks. In: iTextbooks@AIED (2019)
4. Brusilovsky, P.: Webex: Learning from examples in a programming course. In: WebNet (2001)
5. Brusilovsky, P., Edwards, S., Kumar, A., Malmi, L., Benotti, L., Buck, D., Ihantola, P., Prince, R., Sirkiä, T., Sosnovsky, S., Urquiza, J., Vihavainen, A., Wollowski, M.: Increasing adoption of smart learning content for computer science education. In: Proceedings of the Working Group Reports of the 2014 on Innovation amp; Technology in Computer Science Education Conference. p. 31–57. ITiCSE-WGR '14, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2713609.2713611
6. Brusilovsky, P., Malmi, L., Hosseini, R., Guerra, J., Sirkiä, T., Pollari-Malmi, K.: An integrated practice system for learning programming in python: design and evaluation. Research and Practice in Technology Enhanced Learning **13** (2018)
7. Brusilovsky, P., Schwarz, E., Weber, G.: Electronic textbooks on WWW: from static hypertext to interactivity and adaptivity, pp. 255–261. Educational Technology Publications, Englewood Cliffs, New Jersey (1997)
8. Chacon, I.A., Barria-Pineda, J., Akhuseyinoglu, K., Sosnovsky, S.A., Brusilovsky, P.: Integrating textbooks with smart interactive content for learning programming. In: iTextbooks@AIED (2021)
9. Chau, H., Barria-Pineda, J., Brusilovsky, P.: Learning content recommender system for instructors of programming courses. In: AIED (2018)
10. Dresscher, L., Chacon, I.A., Sosnovsky, S.A.: Generation of assessment questions from textbooks enriched with knowledge models. In: iTextbooks@AIED (2021)
11. Ericson, B.J., Guzdial, M.J., Morrison, B.B.: Analysis of interactive features designed to enhance learning in an ebook. In: Proceedings of the 11th International Conference on International Computing Education Research. ACM (aug 2015). https://doi.org/10.1145/2787622.2787731,
12. Ericson, B.J., Miller, B.N.: Runestone: A platform for free, on-line, and interactive ebooks. In: Proceedings of the 51st ACM Technical Symposium on Computer Science Education. p. 1012–1018. Association for Computing Machinery, New York, NY, USA (2020), https://doi.org/10.1145/3328778.3366950
13. Fouh, E., Karavirta, V., Breakiron, D.A., Hamouda, S., Hall, S., Naps, T.L., Shaffer, C.A.: Design and architecture of an interactive etextbook – the OpenDSA system. Science of Computer Programming **88**, 22–40 (2014). https://doi.org/https://doi.org/10.1016/j.scico.2013.11.040

14. Hosseini, R., Akhuseyinoglu, K., Brusilovsky, P., Malmi, L., Pollari-Malmi, K., Schunn, C.D., Sirkiä, T.: Improving engagement in program construction examples for learning python programming. International Journal of Artificial Intelligence in Education **30**, 299–336 (2020)
15. Hsiao, I., Sosnovsky, S.A., Brusilovsky, P.: Guiding students to the right questions: adaptive navigation support in an e-learning system for java programming. J. Comput. Assist. Learn. **26**, 270–283 (2010)
16. Huang, A., Hancock, D., Clemson, M., Yeo, G.C., Harney, D.J., Denny, P., Denyer, G.: Selecting student-authored questions for summative assessments. bioRxiv (2020)
17. Ihantola, P., Karavirta, V.: Two-dimensional parson's puzzles: The concept, tools, and first observations. Journal of Information Technology Education **10**, 119–132 (2011), https://jite.org/documents/Vol10/JITEv10IIPp119-132Ihantola944.pdf
18. Karavirta, V., Ihantola, P., Koskinen, T.: Service-oriented approach to improve interoperability of e-learning systems. In: 2013 IEEE 13th International Conference on Advanced Learning Technologies. pp. 341–345 (2013). https://doi.org/10.1109/ICALT.2013.105
19. Koc-Januchta, M.M., Schönborn, K.J., Tibell, L.A.E., Chaudhri, V.K., Heller, H.C.: Engaging with biology by asking questions: Investigating students' interaction and learning with an artificial intelligence-enriched textbook. Journal of Educational Computing Research **58**, 1190 – 1224 (2020)
20. Ni, L., Bao, Q., Li, X., Qi, Q., Denny, P., Warren, J., Witbrock, M., Liu, J.: Deepqr: Neural-based quality ratings for learnersourced multiple-choice questions. ArXiv **abs/2111.10058** (2021)
21. Pollari-Malmi, K., Guerra, J., Brusilovsky, P., Malmi, L., Sirkiä, T.: On the value of using an interactive electronic textbook in an introductory programming course. In: Proceedings of the 17th Koli Calling International Conference on Computing Education Research. p. 168–172. Koli Calling '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3141880.3141890, https://doi.org/10.1145/3141880.3141890
22. Rahdari, B., Brusilovsky, P., Thaker, K., Barria-Pineda, J.: Knowledge-driven wikipedia article recommendation for electronic textbooks. In: EC-TEL (2020)
23. Ritter, S., Fisher, J., Lewis, A., Finocchi, S.B., Hausmann, B., Fancsali, S.: What's a textbook? envisioning the 21st century k-12 text. In: iTextbooks@AIED (2019)
24. Rößling, G., Naps, T., Hall, M., Karavirta, V., Kerren, A., Leska, C., Moreno, A., Oechsle, R., Rodger, S.H., Urquiza-Fuentes, J., Velázquez-Iturbide, J.: Merging interactive visualizations with hypertextbooks and course management. SIGCSE Bull. **38**(4), 166–181 (2006). https://doi.org/10.1145/1189136.1189184
25. Sirkiä, T.: Jsvee kelmu: Creating and tailoring program animations for computing education. Journal of Software: Evolution and Process **30**(2) (2018), https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.1924
26. Sosnovsky, S.A., Brusilovsky, P.: Evaluation of topic-based adaptation and student modeling in quizguide. User Modeling and User-Adapted Interaction **25**, 371–424 (2015)
27. Thaker, K.M., Brusilovsky, P., He, D.: Concept enhanced content representation for linking educational resources. In: 2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI). pp. 413–420 (2018). https://doi.org/10.1109/WI.2018.00-59
28. Vihavainen, A., Vikberg, T., Luukkainen, M., Pärtel, M.: Scaffolding students' learning using test my code. In: Proceedings of the 18th ACM Conference

on Innovation and Technology in Computer Science Education. p. 117–122. ITiCSE '13, Association for Computing Machinery, New York, NY, USA (2013). https://doi.org/10.1145/2462476.2462501
29. Walker, E., Wong, A., Fialko, S., Restrepo, M.A., Glenberg, A.M.: Embrace: Applying cognitive tutor principles to reading comprehension. In: International Conference on Artificial Intelligence in Education. pp. 578–581. Springer (2017)
30. Walker, E., Wylie, R., Danielescu, A., Rodriguez III, J.P., Finn, E.: Balancing student needs and learning theory in a social interactive postdigital textbook. In: End-user considerations in educational technology design, pp. 141–159. IGI Global (2018)
31. Weber, G., Brusilovsky, P.: ELM-ART: An adaptive versatile system for web-based instruction. International Journal of Artificial Intelligence in Education **12**(4), 351–384 (2001)
32. Xu, Y., Warschauer, M.: Exploring young children's engagement in joint reading with a conversational agent. In: Proceedings of the Interaction Design and Children Conference. p. 216–228. IDC '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3392063.3394417
33. Zingaro, D., Cherenkova, Y., Karpova, O., Petersen, A.: Facilitating code-writing in PI classes. In: Proceeding of the 44th ACM Technical Symposium on Computer Science Education. p. 585–590. SIGCSE '13, Association for Computing Machinery, New York, NY, USA (2013)