# Transformation of PDF Textbooks into Intelligent Educational Resources

Isaac Alpizar-Chacon[0000−0002−6931−9787], Max van der Hart, Zef S. Wiersma, Lorenzo S.J. Theunissen, and Sergey Sosnovsky[0000−0001−8023−1770]

Utrecht University, Utrecht, The Netherlands
i.alpizarchacon@uu.nl, m.vanderhart@students.uu.nl,
z.s.wiersma@students.uu.nl,
l.s.j.theunissen@students.uu.nl,s.a.sosnovsky@uu.nl

**Abstract.** The paper presents Intextbooks - the system for automated conversion of PDF-based textbooks into intelligent educational Web resources. The papers focuses on the new component of Intextbooks responsible for transformation of PDF-based content into semantically-annotated HTML/CSS. The architecture of the system, the design of the client application rendering resulting textbooks and a short validation experiment demonstrating the quality of the transformation workflow are presented.

**Keywords:** Digital textbooks · Interactive textbooks · Intelligent textbooks · Adaptive textbooks · Modeling textbooks · Educational resources · PDF to HTML.



## 1 Introduction

[1]Digital textbooks have become a standard medium for distributing educational content, especially, online. The popularity of digital textbooks has been on the rise. For example, according to DeNoyelles's and Raible's report, the number of students that used digital textbooks at least once in their college studies increased from 44% in 2012 to 66% in 2016 [11]. Some of these textbooks come equipped with additional intelligent services built around them to support enhanced search [24], easier navigation [5], interactive content [14], and, ultimately, better learning [23]. However, most digital textbooks exist as mere digital copies of their printed counterparts. One of the reasons for an insufficient number of intelligent textbooks is the amount of efforts and expertise necessary to create them. Linking relevant parts of a textbook content to each other, to external interactive resources and to the elements of domain knowledge are tasks that traditionally require manual input from domain and pedagogy experts, thus preventing development and deployment of intelligent textbooks at scale.

In our recent work, we have made initial steps to address this problem by developing an approach towards automated extraction of machine-readable domain models from PDF textbooks. This approach takes into account common patterns of textbook formatting and organization and formalizes them as an extensible set of rules that gradually process the raw content of a textbook, elicit its underlying structural and formatting elements and, finally, harvest the semantic model of a textbook [3]. We focus mainly on PDF as the most common (and more challenging) format for distributing digital textbooks. Additionally, we have used such models to create semantic bridges allowing us to connect the textbooks to one another and to DBpedia [1]. This opens up a range of interesting possibilities for supporting development of intelligent and adaptive textbooks at scale. One potential application of this approach has been presented in [2], where we have implemented the Interlingua application capable to link sections of relevant textbooks across multiple languages to support students studying in a foreign tongue. However, while our approach is capable to extract a semantic model of a textbook, link it to other models and essentially provide a backbone for implementing intelligent services, it has been missing an important component that performs the actual conversion of static PDF files into interactive resources that can be published on the Web.

In this paper, we fill this gap and present Intextbooks (Intelligent textbooks) - the system that can perform the complete transformation of PDF textbooks into online intelligent educational resources. After extracting a semantic model from a PDF textbook, it converts it into an HTML/CSS representation with an enriched fine-grained DOM (Document Object Model)[2]. Every content/layout/structure element of the textbook (words, lines, paragraphs, pages, chapters, index entries, etc.) are uniquely identifiable within its DOM. As a result, this implementation is absolutely flexible in terms of potential interactivity as virtually any object of a textbook (from a chapter to a keyword) can become an object of targeted interaction. Moreover, each and every element of a textbook's DOM is also identifiable within the semantic model extracted from the textbook. As a result, the entire textbook becomes an integrated resource where content elements and pieces of domain knowledge are interlinked on both the presentation and the knowledge levels. Such an organization enables various kinds of semantically- and adaptively-enhanced interaction with the textbook content.

The rest of this paper is structured as follows. Section 2 gives an overview of related work. Section 3 provides the details of the Intextbooks system. Section 4 presents a validation experiment conducted to test the PDF to HTML conversion component of the system. Finally, section 5 concludes the paper with a discussion and a description of future work.

[2] https://www.w3.org/DOM/DOMTR

## 2 Related work

### 2.1 PDF to HTML conversion

Basic conversion from PDF to HTML documents is a well-known task. In 2003, Rahman and Alam [22] discussed the use of Document Image Analysis techniques to identify the layout and basic components (graphics, text, tables) of PDF documents as the first step to produce HTML outputs. Marinai, Marino and Soda [20] presented a system for converting PDF books to the ePub format, which produces XHTML[3] files. They focused on TOC (TOC) identification and analysis, as well as identifications of notes and illustrations. Those two analyses allow for generating more structured XHTML files. Currently, there are multiple online tools[4] and libraries[5] that allows easy and quick PDF to HTML conversion. The main limitation of existing tools is that they only focus on producing an HTML with an accurate visual representation of the original PDF document without any mechanism that preserves semantic information about its content or structure.

### 2.2 Interaction with textbooks

A good interaction design is an important factor affecting user acceptance of digital textbooks [8]. There are multiple ways in which a digital textbook can support a reader's interaction with its content. They range from more standard tools such as textual search and internal hyperlinks to social interactive interfaces such as bookmarking, tagging and commenting [12, 15, 16, 19], to intelligent services such as semantic search [13] and adaptive navigation support [4]. Semantically-enriched textbooks are capable to support meaningful linking and rearrangement of content [17], semantic search and retrieval of relevant learning objects [21] and even targeted inquiry, exploration and comparison of important notions in the domain [7]. Adaptive textbooks trace progress of their readers and maintain composite models of their knowledge to provide personalized content or interaction. For example, in [5], an online textbook with adaptive navigation support annotates links to its resources with indicators to inform students about the individual educational value of the linked resources. Other examples of adaptation can be found in [18, 9, 25].

## 3 The Intextbooks system

In this section we describe Intextbooks, a system that transforms PDF textbooks into interactive intelligent educational resources.

[3] https://www.w3.org/TR/xhtml1/
[4] https://www.pdf2html.org/, https://pdf.io/pdf2html/, etc.
[5] see Section 3.2

### 3.1 Architecture

The Intextbooks system consists of two main groups of components. The offline components perform the tasks of textbook modeling and conversion to HTML, while the online components support students' interaction with the textbooks. Figure 1 presents the overall architecture of the system.

The offline components take a PDF textbook and, first, extract its semantic model. The resulting model is represented as an RDF-enriched TEI (Text Encoding Initiative)[6] document. Each word, line, text fragment, page, (sub)section, index term, TOC entry, etc. is recognized individually in this model. Then, the PDF textbook is converted into an HTML representation. As the last step, TEI and HTML representations are synchronized, meaning all elements of the TEI model are connected to the DOM elements of the HTML version of the textbook. After that, both the TEI and the synchronized HTML representation of the textbook are stored in the central repository. Together, they play the roles of domain and content models to enable semantic and adaptive access to the textbook content.

The online Web-reader presents processed textbooks to students. Every time a student requests a textbook, the reader displays the synchronized HTML representation of the textbook and supports various kinds of interaction with it. Additionally, the adaptation engine uses a student model and activity logs to generate specific content and interactions for each student (e.g., tailored navigational aid). Thanks to the extracted model of the textbook, the web interface is aware of various elements of the textbook semantics: the precise beginning and ending of each (sub)section, the relevant terms in every page (based on the index terms) and additional information associated with them (definitions, links to external resources), etc.

### 3.2 Main components

**Textbook model extractor** Our textbook model extractor is a rule-based component that first extracts a semantic model of a textbook using its symbolic, formatting, and structural elements. Then, the model is enriched with additional semantic information using DBpedia and the identified index terms in the textbook. Finally, the enriched semantic model is serialized as an TEI/XML file. Altogether, 55 unique rules have been defined to handle the extraction process. The implementation details of this approach are described in our previous work [3, 1].

The resulting TEI textbook model groups all the information extracted from the textbook into three categories: structure, content, and domain knowledge. The structure section contains the hierarchical structure of (sub)sections of the textbook, according to the TOC. The content section provides for every section its textual content organized according to basic textual elements: words, lines, fragments, pages, and subsections. All these elements have unique IDs. Finally,

[6] https://tei-c.org/

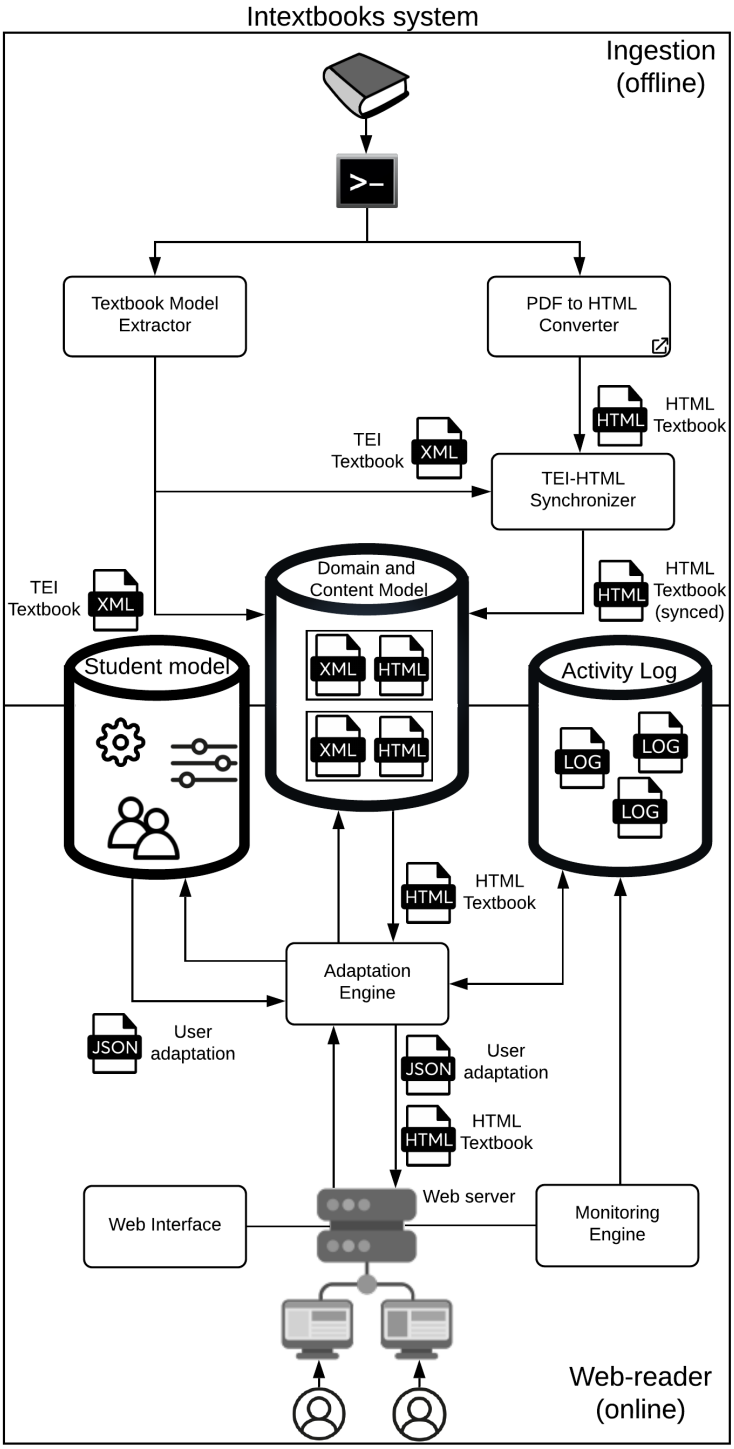


**Fig. 1.** Intextbooks architecture

the domain knowledge section provides important terms in the textbook according to its Index. Each term is linked to the (sub)sections and pages where it appears, and it can have additional semantic information such as definition, external resource, classification categories, and related terms. This semantic information is incorporated into the TEI model using RDFa[7] attributes, which create connections to the Linked Open Data cloud[8].

**PDF to HTML converter** This component converts a PDF textbook into an HTML representation that preserves its layout and content. Preserving the layout of a PDF document is a complex task. A PDF document contains thousands of low-level objects grouped into three layers – text, bitmap images, and vector graphics – [6]). Several open libraries have been developed to perform the low-level processing of these PDF primitives and converting them into an HTML representation. We considered four such libraries: pdf2htmlEX[9], PDFMiner[10], pdf2html[11], and Xpdf[12].

While comparing possible libraries, we looked for different properties. The most important was the ability of the conversion library to preserve the look of the PDF in the resulting HTML. Therefore, our search was mostly limited to geometrically-based conversion libraries. Another critical factor was the ability to parse the HTML in a structured order to synchronize the HTML with the textbook semantic model. Other factors that were considered were Linux support, performance, and scalability. After analyzing the candidate libraries, we have decided to use pdf2htmlEX.

This library is no longer under active development, but it has an extensive documentation allowing its enrichment. It preserves the layout of the PDF textbook perfectly across different types of documents. It can be ran as a standalone process and is quite fast compared to other tools. Furthermore, the HTML output is structured: each page has its own id and is divided into lines. One downside of the library is that tables, graphs, figures, and vector lines are all grouped into one static background image that is loaded for an entire page, which makes it harder to recognize individual elements.

**TEI-HTML synchronizer** This component modifies the output HTML representation of a textbook to create a fine-grained DOM structure, which is synchronized to the textual elements from the semantic model of the textbook. This process is not straightforward since HTML's structure exists to preserve the layout, but it does not correspond to textual or semantic elements. For example, in the HTML produced by the conversion library, multiple words may share a single *span* element. After the synchronization process, the final HTML for a textbook

[7] http://rdfa.info/
[8] https://lod-cloud.net/
[9] https://github.com/coolwanglu/pdf2htmlEX
[10] https://pypi.org/project/pdfminer/
[11] https://github.com/mgedmin/pdf2html
[12] https://www.xpdfreader.com/pdftohtml-man.html

has a DOM structure that identifies words, lines, and paragraphs with the same IDs from their corresponding elements in the TEI document. The rest of this subsection explains our matching algorithm to synchronize both representations of a textbook.

*Internal Representation.* The matching takes as an input internal representations for both the TEI model and the HTML file. These representations consist of lists of pages, each of which contains lines (for the HTML) or paragraphs with lines (for the TEI), which themselves contain lists of words. For HTML lines, we also keep extra information, such as their X- and Y-position on a page and font size, which are necessary for the consequent matching. For the words, we keep track of whether they have already been matched. It is important to note that the HTML text is split into DOM's *TextNodes*. These may contain characters from different words (split by a space), parts of a single word, an entire word, or only whitespaces. Different parts of a word in the HTML might also be at different levels, split by *spans*. Therefore, we keep a list of the *TextNodes* that all together form a word in the HTML according to the TEI model.

*Matching Words.* After the internal representation has been built, the algorithm matches the words between the TEI and HTML representations of a textbook. The difficulty comes from the fact that the HTML produced by pdf2htmlEX does not preserve the structure of words. This is also the most important part since the matching of lines and paragraphs depends on this initial matching. For each page, words are matched separately. First, the entire text of a page is extracted separately from the TEI and HTML representations. Then, both texts are compared using the *Google's Diff Match Patch* library[13]. This library compares the words and determines which words in the HTML belong to which words in the TEI representation. The result is a list containing differences and matches between the text of both pages. Finally, for each matched TEI word, the HTML is updated to wrap the matching *TextNodes* as a DOM element with the ID corresponding to the matched word.

There are some special cases when matching words. The first one is when there is no one-to-one relation between the words from the TEI and HTML representations. When a word in the TEI model corresponds to multiple elements in the HTML, we can match the words by aggregating corresponding elements in the HTML and giving them the same ID. The opposite case is more complex. When an individual element in the HTML corresponds to multiple words in the TEI, the *TextNodes* for the HTML element are to be split based on the words in the TEI. Then, the *TextNodes* that contain the characters for each TEI word are wrapped together with the ID of the respective word. Another case occurs when the pdf2htmlEX library inserts special characters that are different from the characters in the TEI representation. For example, sometimes the HTML contains orthographic ligatures, rather than Latin characters. The TEI representation contains no ligatures; therefore, in a preprocessing step, we replace

[13] https://github.com/google/diff-match-patch

some special Unicode characters by their correct counterparts. The replacement list is stored is a separate file; we extend it as we observe new cases.

*Matching Lines and Paragraphs.* After the words have been matched, the algorithm starts matching the lines and paragraphs. This process is quite simple using the already matched words. For each line in the TEI representation, the algorithm finds the corresponding words of that line in the HTML, gets their parent element, and wraps it with the right ID. To further improve accuracy, a cross-check is carried out after the line matching. If an HTML line did not get a match, but if the previous and next line did and they have the same ID, the algorithm also wraps the current line with the same ID. A similar process is done to match paragraphs, but in this case, the algorithm keeps track of matched lines in the HTML to detect and wrap the elements that represent paragraphs.

### 3.3 Web interface

The web interface of the Intextbooks system will allow students to engage and interact with the textbooks in several ways. Currently, we are working on the design and development of this and the other components of the system, such as the student model. Figure 2 presents the working prototype of the web interface.

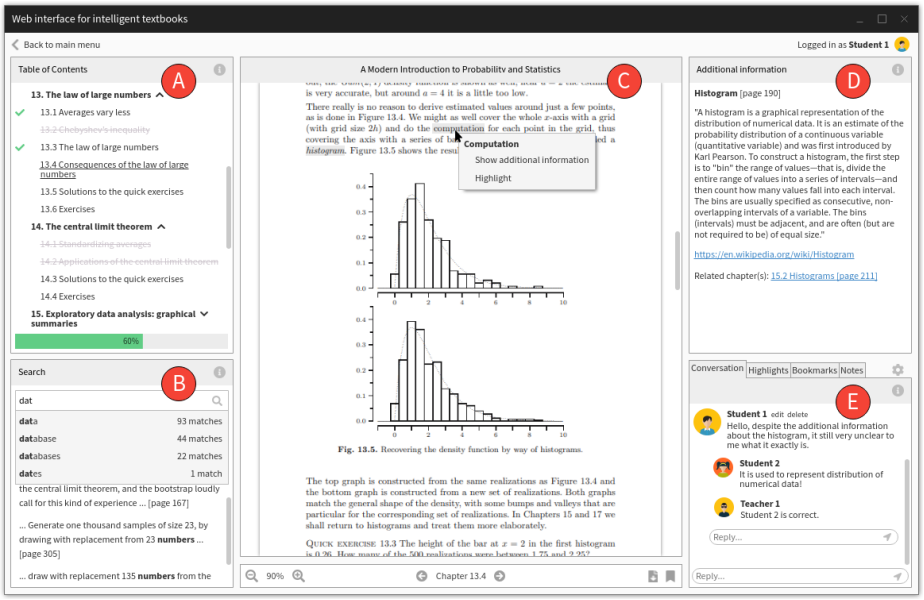


**Fig. 2.** Intextbooks web interface prototype

The web interface is divided into three parts. The main one is shown in the middle; this is where the textbook is displayed. There are smaller panel on either

side of the main part. They provide additional tools for the user to interact with the textbook. Two buttons on the top of the screen can be used to navigate to the main menu and user settings.

The web interface provides a TOC to the reader (panel A in Figure 2). The TOC contains a reference to each (sub)section. The user can click on one of the entries, and the web interface will show to the corresponding content of the section the middle panel. Furthermore, when the teacher sets a path for a specific textbook or an adapted path is generated for a user, this will be reflected in the TOC. The TOC entry that are not included in the path will be crossed out and have a lighter color. Lastly, the TOC displays annotations in the form of checkmarks and a progress bar to provide navigational cues to the user.

The other feature in the left part of the web interface is the search tool (panel B). When searching a word, the web interface will suggest possible matches prioritizing important terms from the textbook model. The number of matches is also included in the suggestions. After a search, the web interface will show snippets of text where a keyword occurs. The user can make a more precise decision based on these snippets. When clicked on a snippet, the web interface will browse to the corresponding item in the textbook and highlight the corresponding searched term.

An important aspect of creating an intelligent textbook is having interaction with the textbook. The web interface provides interaction by allowing the user to click on certain words that are highlighted in the text. When the user left-clicks on a highlighted word, additional information will be shown in a panel on the right side of the web interface. The additional information can contain definitions, links, or references to related chapters. This information also comes from the semantic model of the textbook. Panel D in Figure 2 shows the additional information for the *histogram* term.

Other features are grouped under the same panel using tabs (panel E in Figure 2). When clicked on either tab, the web interface will show the corresponding tool. The different tools can be enabled/disabled using the settings button. For the moment, we define four tools, but more can be added as necessary. The first tool is used to interact with other students or teachers. Questions can be asked here to be answered by other students or teachers. The second tool is used to highlight text in the textbook. Different colors can be used to categorize highlighted text. The third tool is used to create bookmarks. Bookmarks can be used to improve navigation among different pages. Lastly, the user can create notes in a simple text editor, similar to the text editor that most operating systems provide.

The most important component of an textbook to interact with is the its actual content. The textbook is displayed in the middle part of the interface, along with extra options at the bottom (panel C). The user can zoom in and out, skip to the next chapter, download the content as a PDF-file, or bookmark the current page. The user can also right-click on the highlighted terms to display a context menu with additional actions. As our system grows, this part of the interface can provide multiple ways for the users to interact with the textbooks. For example,

tables could become interactive elements where the user can change the order of the data or create aggregations. Interaction with fine-grained elements, such as words, is possible thanks to the creation of the identifiable DOM structure in each HTML representation. It is also worth mentioning that the web interface is being developed for both desktop and smartphone displays.

**Monitoring Engine** The web interface communicates directly to the monitoring engine to log every action of a student. All the logs are store in a activity log repository.

**Adaptation engine and student model** These components are planned to be added in the future as the Intextbooks system matures and starts utilizing the domain extracted from the textbooks and the activity of students.

## 4 Validation

We have conducted a validation experiment to test the accuracy of the matching algorithm, which is used to create the fine-grained identifiable DOM structure in the HTML textbooks. This DOM structure coupled with the extracted TEI model is required to offer the capability of fine-grained, flexible, semantically-enriched interactions with textbooks.

### 4.1 Procedure

We have used a test set of 70 university-level textbooks in several domains: statistics, computer science, web programming, literature, and history. All textbooks are written in English. To estimate the accuracy of the matching algorithm, we have used the percentage of words that were matched between the TEI and HTML representations of each book as our evaluation metric. Such a metric is a good indication for the accuracy of the algorithm since the word matching is the most challenging part, and the line and paragraph matching depend on the number of matched words.

We have run the validation using thee variations of the matching algorithm. Currently, the matching of words is not 100% correct, because sometimes the order of words in the generated HTML is different from the one in the extracted TEI model, and because subscripts and superscripts in the HTML are not always placed in the correct position. Therefore, we have implemented a version of the matching algorithm that uses a threshold to merge superscripts and subscripts with either the previous or next line to try to increase matching accuracy. This variant of the algorithm also sorts the words in the HTML based on their Y-position, so they are read in the same order as in the TEI representation. For the validation, we have tested the accuracy of the original matching algorithm (no threshold), an algorithm with a fixed threshold, and a variant with a dynamic threshold based on the most frequent distance between two lines on a page. Table 1 shows the results for the validation using the test set and the three algorithms.

| | No Threshold | Fixed Threshold | Dynamic Threshold |
|---|---|---|---|
| Mean | 87.16 | 88.76 | 87.09 |
| Median | 90.53 | 91.15 | 88.63 |
| Standard Deviation | 12.45 | 12.22 | 13.15 |

**Table 1.** Validation of the matching algorithms.

### 4.2 Analysis

The results show that at least 87% of all the words are matched for all three methods. The obtained values indicate that the matching algorithm requires some adjustments to match the remainder of the words.

Textbooks that mostly consist of text (without other elements such as formulae and tables) get a near 100% matching rate. However, the obtained values are mostly determined by more complex textbooks. Such textbooks get a higher mismatching rate because of the figures, tables, graphs, and other elements that are represented as text in the TEI model. However, they are converted to images by the pdf2htmlEX library. As a result, these elements reduce the matching rate. For example, one of the textbooks [10], has a mismatch rate of about 15% due to such discrepancies. Subscripts and superscripts in text also considerably reduce matching rates. Using only the distance between the previous and next lines does not provide a reliable indicator for whether an element is a superscript or a subscript. A possible solution is to look at the word that gets formed by adding the subscript/superscript to both the previous and next line, and see which of these two words occurs in the TEI for that page. If the threshold algorithm is further extended and improved so that the HTML and TEI representations are read in the same order, we should be able to get an accuracy approaching the 100% mark.

## 5 Discussion and future work

We have presented the current state of Intextbooks - a system capable of transforming PDF textbooks into interactive and intelligent educational resources. Currently, we can extract high-quality semantic models of the textbooks, and create HTML representations of the same textbooks that are connected to their semantic models using fine-grained DOM structures. We can match around 88% of all the words in the textbooks to individual elements in the HTML resources, and are working to further improve the matching algorithm. We have designed a web interface that allows to interact with the HTML textbooks in multiple ways. We have plans to extend it with adaptive functionality.

There might be several possible concerns both practical and pragmatic about the scale and applicability of this kind of a system. One question that needs to be addressed is the availability of textbooks and the copyrights issues. In our experience, university libraries can supply enough PDF-based textbooks

on a variety of subjects. From the point of copyright protection, if a system provides enhanced access to these books but only to the students of the university holding necessary subscriptions, then publishers do not have a reason to object. In the worst case scenario, many good-quality textbooks are freely available online nowadays in open repositories such as Openstax[14]/Connections[15], Open Textbook Library[16], OER-Commons[17], etc.

We plan to extend further and improve the components of the system. In particular, it is important to better define the semantics of knowledge that is extracted from the textbooks. Using the index terms from the textbooks to precisely identify the sub-domains that are covered in each (sub)section will allow to recommend content to students better. Additionally, we need to test how the automated textbook modeling technology will cope with different textbook formatting across less formal domains (e.g., sociology) or domains with conflicting viewpoints (e.g., history).

For this system to be as complete as we want, we need to add the missing components to infer the current state of students' knowledge and provide meaningful adaptation. Another direction for future work is to research on how to produce fully interactive HTML elements from the static tables, charts, and formulae in the textbooks. Finally, it is important to evaluate the system's effectiveness in a user study with real students from a target group. Both the user experience and the possible effect on learning need to be investigated.

## References

1. Alpizar-Chacon, I., Sosnovsky, S.: Expanding the web of knowledge: one textbook at a time. In: Proceedings of the 30th ACM Conference on Hypertext and Social Media. ACM, New York, NY, USA (2019)
2. Alpizar-Chacon, I., Sosnovsky, S.: Interlingua: Linking textbooks across different languages. In: Proceedings of the First Workshop on Intelligent Textbooks. vol. 2384, pp. 104–117. CEUR-WS (2019)
3. Alpizar-Chacon, I., Sosnovsky, S.: Order out of chaos: Construction of knowledge models from pdf textbooks. In: Proceedings of the 20th ACM Symposium on Document Engineering (Accepted). DocEng 2020, ACM, New York, NY, USA (2020)
4. Brusilovsky, P.: Adaptive navigation support. In: The adaptive web, pp. 263–290. Springer (2007)
5. Brusilovsky, P., Eklund, J.: A study of user model based link annotation in educational hypermedia. Journal of Universal Computer Science **4**(4), 429–448 (1998)
6. Chao, H., Fan, J.: Layout and Content Extraction for PDF Documents. In: Document Analysis Systems VI, vol. 3163, pp. 213–224 (2004)
7. Chaudhri, V.K., Cheng, B., Overtholtzer, A., Roschelle, J., Spaulding, A., Clark, P., Greaves, M., Gunning, D.: Inquire biology: A textbook that answers questions. AI Magazine **34**(3), 55–72 (2013)

[14] https://openstax.org
[15] https://cnx.org/
[16] https://open.umn.edu/opentextbooks
[17] https://www.oercommons.org/

8. Chiu, T.K.F.: Introducing electronic textbooks as daily-use technology in schools: A top-down adoption process. British Journal of Educational Technology **48**(2), 524–537 (2017). https://doi.org/10.1111/bjet.12432, https://onlinelibrary.wiley.com/doi/abs/10.1111/bjet.12432
9. De Bra, P.M.: Teaching through adaptive hypertext on the www. International Journal of Educational Telecommunications **3**(2), 163–179 (1997)
10. Dekking, F.M., Kraaikamp, C., P., L.H., Meester, L.E.: A modern introduction to probability and statistics: understanding why and how. Springer (2005)
11. DeNoyelles, A., Raible, J.: Exploring the use of e-textbooks in higher education: A multiyear study. Educause Review. Retrieved from https://er. educause. edu/articles/2017/10/exploring-the-use-of-e-textbooks-in-higher-education-a-multiyear-study (2017)
12. Di Vesta, F.J., Gray, G.S.: Listening and note taking. Journal of educational psychology **63**(1), 8 (1972)
13. Dichev, C., Dicheva, D.: View-based semantic search and browsing. In: 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06). pp. 919–925. IEEE (2006)
14. Ericson, B.: An analysis of interactive feature use in two ebooks pp. 1–14 (2019)
15. Fisher, J.L., Harris, M.B.: Effect of note taking and review on recall. Journal of Educational Psychology **65**(3), 321 (1973)
16. Gall, M.D.: The use of questions in teaching. Review of educational research **40**(5), 707–721 (1970)
17. Glushko, R.J.: The discipline of organizing. BULLETIN of the American society for information science and technology **40**(1), 21–27 (2013)
18. Kavcic, A.: Fuzzy user modeling for adaptation in educational hypermedia. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **34**(4), 439–449 (2004)
19. Kissinger, J.: The social & mobile learning experiences of students using mobile e-books. Online Learning **17**(1) (2013). https://doi.org/10.24059/olj.v17i1.303, https://olj.onlinelearningconsortium.org/index.php/olj/article/view/303
20. Marinai, S., Marino, E., Soda, G.: Conversion of pdf books in epub format. In: 2011 International Conference on Document Analysis and Recognition. pp. 478–482. IEEE (2011)
21. Melis, E., Goguadze, G., Homik, M., Libbrecht, P., Ullrich, C., Winterstein, S.: Semantic-aware components and services of activemath. British Journal of Educational Technology **37**(3), 405–423 (2006)
22. Rahman, F., Alam, H.: Conversion of pdf documents into html: a case study of document image analysis. In: The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003. vol. 1, pp. 87–91. IEEE (2003)
23. Ritter, S., F.J.L.A.B.F.S.H.B..F.S.: What's a textbook? envisioning the 21st century k-12 pp. 87–94 (2019)
24. Sosnovsky, S., D.M.A.E.G.G.W.S.L.P.S.J..M.E.: Math-bridge: Closing gaps in european remedial mathematics with technology-enhanced learning. In: Mit werkzeugen mathematik Und stochastik lernen–using tools for learning mathematics and statistics, pp. 437–451. Springer (2014)
25. Ullrich, C., Melis, E.: Pedagogically founded courseware generation based on htn-planning. Expert Systems with Applications **36**(5), 9319–9332 (2009)