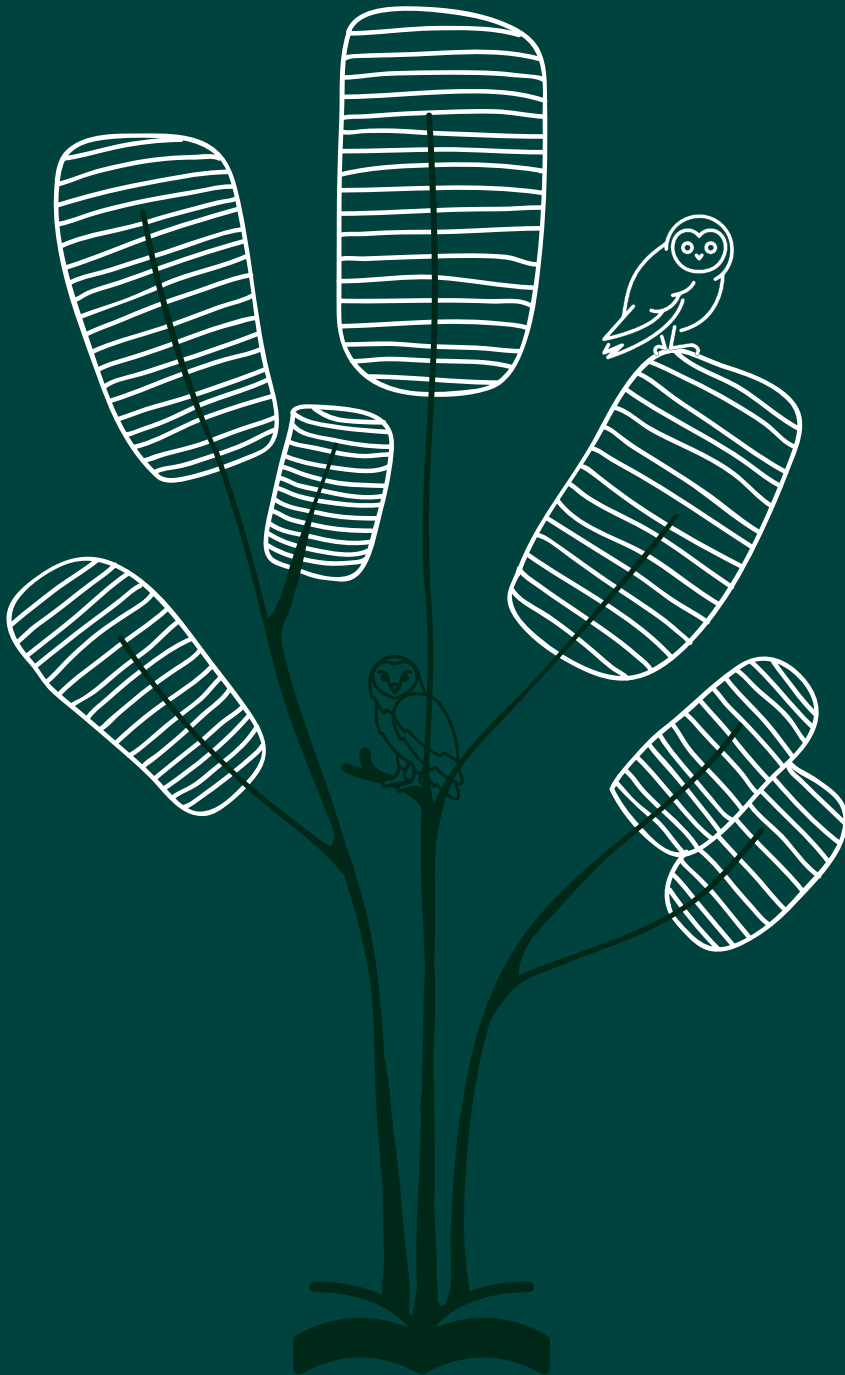


Extraction of Knowledge Models from Textbooks

Isaac Alpizar Chacon



Extraction of Knowledge Models from Textbooks

Isaac Alpizar Chacon

Printed by proefschriftenprinten.nl
Cover design by Isaura Ramírez Brenes
DOI <https://doi.org/10.33540/1647>

Department of Information and Computing Sciences, Utrecht University
Copyright © 2023 by Isaac Alpizar Chacon

Extraction of Knowledge Models from Textbooks

Extractie van Kennismodellen uit Studietoeken
(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor aan de
Universiteit Utrecht
op gezag van de
rector magnificus, prof.dr. H.R.B.M. Kummeling,
ingevolge het besluit van het college voor promoties
in het openbaar te verdedigen op

woensdag 8 maart 2023 des middags te 4.15 uur

door

Isaac Alpizar Chacon

geboren op 07 april 1987
te Alajuela, Costa Rica

Promotor:

Prof.dr. J.T. Jeuring

Copromotor:

Dr. S.A. Sosnovsky

This thesis was accomplished with financial support from *Ministerio de Ciencia, Innovación, Tecnología y Telecomunicaciones* (MICITT), Costa Rica (Grant No: 2-1-4-17-1-021).

Contents

	Page
Acronyms	1
1 Introduction	3
1.1 The Problem and Motivation	4
1.2 Main Elements of Textbooks	6
1.3 Extraction of Knowledge Models	8
1.4 Research Questions	12
1.5 Thesis Structure	12
2 Knowledge Extraction from Unstructured and Semi-Structured Textual Resources	17
2.1 Introduction	19
2.2 Preliminaries	20
2.3 Approaches for Knowledge Extraction from Textual Resources	21
2.4 Other Dimensions	43
2.5 Limitations	44
2.6 Conclusions and Future Work	44
3 Order out of Chaos: Construction of Knowledge Models from PDF Textbooks	47
3.1 Introduction	49
3.2 Related Work	50
3.3 Approach	51
3.4 Evaluation	60
3.5 Conclusions and Future Work	66
4 Expanding the Web of Knowledge: One Textbook at a Time	67
4.1 Introduction	69
4.2 Related Work	70
4.3 Approach	72
4.4 Evaluation	81
4.5 Conclusions and Future Work	89

5 What's in an Index: Extracting Domain-specific Knowledge Graphs from Textbooks	91
5.1 Introduction	93
5.2 Related Work	95
5.3 Preliminaries	96
5.4 Approach	98
5.5 Evaluation	104
5.6 Conclusion and Future Work	110
6 The Power of the Curve: Measuring the Quality of Extracted Concepts Using Learning Curve Analysis	113
6.1 Introduction	115
6.2 Related Work	117
6.3 Experiment	117
6.4 Results and Analysis	125
6.5 Conclusion and Future Work	136
7 Lights, Camera, Action! Applications of Textbook Knowledge Models	137
7.1 Introduction	139
7.2 Interlingua: Linking Textbooks Across Different Languages	140
7.3 InTextbooks: Transforming PDF Textbooks into Interactive Educational Textbooks.	146
7.4 InTextbooks/P ⁴ : Integrating Textbooks with Smart Interactive Content for Learning Programming	155
7.5 Conclusion	164
8 Conclusion	167
8.1 Revisiting the Research Questions	168
8.2 Limitations	170
8.3 Future Work	171
A Learning Curves	173
B Implementation	181
Backmatter	
Bibliography	183
Summary	203
Samenvatting	205
Acknowledgments	209

Curriculum Vitae

213

Acronyms

AI Artificial Intelligence. 4

AIED Artificial Intelligence in Education. 156

AIES Adaptive and Intelligent Educational Systems. 4

AL Alternative Labels. 80

CRF Conditional Random Field. 34

DIKW Data-Information-Knowledge-Wisdom. 20

DOM Document Object Model. 147

EC External Concept. 80

EL Entity Linking. 70

EU European Union. 140

InTextbooks Intelligent Textbooks. 147

ISI Glossary ISI Multilingual Glossary of Statistical Terms. 64

ITS Intelligent Tutoring Systems. 117

KB Knowledge Base. 19

KC Knowledge Component. 115

LDA Latent Dirichlet Allocation. 38

LOD Linked Open Data. 69

MCQ Multiple Choice Questions. 143

MG Mastery Grids. 118

MOOCs Massive Open Online Courses. 23

NDCG Normalized Discounted Cumulative Gain. 64

NED Named Entity Disambiguation. 70

NER Named Entity Recognition. 70

NLP Natural Language Processing. 23

OCR Optical Character Recognition. 19

OER Open Educational Resources. 141

P⁴ Python Programming Personalized Practice System. 157

PDF Portable Document Format. 5

PF Preferred Label. 80

QuizPET Quizzes for Python Educational Testing. 118

SVM Support Vector Machine. 34

TEI Text Encoding Initiative. 60

TKM Textbook Knowledge Model. 22

ToC Table of Contents. 7

VSM Vector Space Model. 37

WSD Word Sense Disambiguation. 70

WWW World Wide Web. 41

CHAPTER 1

Introduction

1.1 The Problem and Motivation

Many modern information systems revolve around knowledge, and educational systems are no exception. Adaptive and Intelligent Educational Systems (AIES)¹ require high-quality knowledge representations for content, domain, and user modeling. However, the acquisition, modeling, sharing, and reuse of formally represented knowledge remains the primary bottleneck hindering large-scale deployment and scalability investigation of AIES. Other Artificial Intelligence (AI) applications, like expert or learning-based systems, suffer the same problem.

Acquiring knowledge in the correct format is considered the most lengthy and complex process in developing systems requiring knowledge [219]. Typically, the knowledge is elicited from human experts using various techniques [242]. However, there are multiple drawbacks to this approach [103, 141, 205]. First, the process is highly laborious and time-consuming. Second, experts often have difficulty articulating their knowledge. Finally, experts tend to disagree.

Due to the cost and difficulties associated with the manual creation of models, automated knowledge acquisition techniques have become an important subject. During the '80s, knowledge acquisition research was boosted thanks to the development of expert systems [104]. In the '90s, Tang et al. [258] had already compiled eight computer-based techniques for acquiring data and knowledge. Text documents are typically used as the source of automated knowledge extraction [98, 124, 176, 228]. However, despite the wide availability of textual resources in all domains, scalable knowledge extraction methods for domain models are still an open problem.

Ontologies have been used as a reusable representation of knowledge in multiple domains. However, methods for creating ontologies are usually domain- and application-dependent [8, 280]. The construction of knowledge models should produce representations where the domain semantics are explicitly encoded but in a scalable way. One example of good scalability is the learning of word embeddings in natural language processing [202]. However, the semantics of the resulting vectors are not explainable.

One path to scalable knowledge extraction is focusing on particular resources that satisfy three conditions: (1) high-quality domain-specific content; (2) high-quality semantic structures (implicit or explicit); and (3) ample availability. Textbooks are one of those resources that can foster the creation of knowledge models.

Textbooks are carefully-designed documents with educational content in a specific field of study. According to Murray & Pérez [206], textbooks are the most universally used instructional resource for content dissemination. Textbooks focus on a narrow and cohesive domain, i.e., they are domain-oriented. Additionally, there are textbooks for almost every possible domain. Their primary purpose is to explain the knowledge in the domain to a novice; however, textbooks support both teachers and pupils [212]. Since they are designed and revised by domain experts, textbooks are a high-quality content source.

¹Adaptive and/or intelligent computer applications that support students and teachers in performing educational activities. Adaptive refers to the functionality to be different for different students according to the individual information of students. The term intelligent means that the systems use artificial intelligence techniques to provide broader and better support for the users [43].

In terms of content and structure, textbooks provide a sequence of topics, comprehensive content coverage, high-quality graphics and photos, tabulated data, exercises, and problems [203]. The Table of Contents and back-of-the-book indices provide content listings. The authors' knowledge expressed in the organization and content of textbooks forms a hidden layer of semantics that can be exploited to construct knowledge models (☛ Section 1.2 details the elements of the textbooks that encode knowledge).

Additionally, digital textbooks (also called electronic textbooks, multimedia books, online interactive books, and e-textbooks [233]) are widely available. The basic digital textbooks are digitized versions of traditional textbooks and, in some cases, contain additional interactive features such as highlighting and note-taking [87]. The Portable Document Format (PDF) has been widely used to distribute digital textbooks. Feldstein et al. [96] reported a study on digital textbook use, where students overwhelmingly chose PDF over other formats. The authors mentioned the ability of almost all computers to render PDF textbooks and the students' familiarity with the format as the reason for the format's popularity. Baker-Eveleth & Stone [24] recognized PDF downloads as a way for students to access e-textbooks. PDF textbooks preserve the original content and formatting, displaying the same across all devices and operating systems. Additionally, they are compact in file size and allow for the integration of various types of content (e.g., text, images, vector graphics, and videos). PDF textbooks are extensively available on the Web. Publishers like Wiley² and Springer³ offer thousands of textbooks in PDF format across a variety of disciplines. Proprietary digital textbooks can usually be accessed through the libraries of educational institutions. Sites like Project Gutenberg⁴ and Internet Archive Books⁵ provide free textbooks legally. In addition, there is a recent trend to produce high-quality, peer-reviewed open textbooks. Open Textbook Library⁶, Libre Texts⁷, OpenStax⁸, and Tu Delft Open Textbooks⁹ offer open PDF textbooks for higher education. Nonetheless, the PDF format has one major disadvantage; extracting the content from the documents is not trivial. A PDF textbook only contains information about the characters, their style, and their position on a page. The textbook's structure, such as headings, paragraphs, and sentences, must be recognized.

To summarize, educational systems (and other AI systems) benefit from formal knowledge representations. However, acquiring high-quality knowledge is still considered one of the main obstacles to developing systems requiring knowledge. Digital textbooks are an excellent source for creating knowledge representations, given their domain-oriented content, structure, and availability. The authors' knowledge encoded in the textbooks' elements that facilitate navigation and understanding of the material can be leveraged to create knowledge models. However, extracting the structure and content from digital (PDF) textbooks is challenging. From this per-

²<https://onlinelibrary.wiley.com/>

³<https://link.springer.com/>

⁴<https://www.gutenberg.org/>

⁵<https://archive.org/details/books>

⁶<https://open.umn.edu/opentextbooks>

⁷<https://libretexts.org/>

⁸<https://openstax.org/>

⁹<https://textbooks.open.tudelft.nl/>

spective, this thesis explores the automatic extraction of knowledge models from digital textbooks.

1.2 Main Elements of Textbooks

Before introducing the proposed approach in this thesis, this section describes the main elements of textbooks that are used to extract a model that contains the knowledge in the textbooks and in the domain.

1.2.1 Content

High-quality content is the main element of textbooks since their content is designed to convey a great deal of information to students [57]. Chambliss & Calfee [50] mention three content aspects of well-designed textbooks: (1) content is organized coherently, connecting the domain structure or the student knowledge; (2) the content reflects the disciplinary domain and the typical student's current knowledge; and (3) content is structured using transition link content (e.g., introductions and conclusions). Mahmood [175] describes additional aspects of high-quality content: the coverage of the contents and objectives aligns to a curriculum policy, vocabulary level is adequate, pictures and illustrations are relevant, there are assessments/exercises to content, and the text reliability is accurate, among others. Additionally, content is organized hierarchically into sections (chapters and subchapters). The order of the sections provides relevant information (e.g., content organized from easy to complex or chronologically). In summary, content from high-quality and well-designed textbooks is a unique source of valuable domain knowledge. Figure 1.1 shows an example of hierarchical content from a textbook: the paragraphs are associated with the "Introduction" subchapter, which in turn is part of the "Descriptive Statistics" chapter.

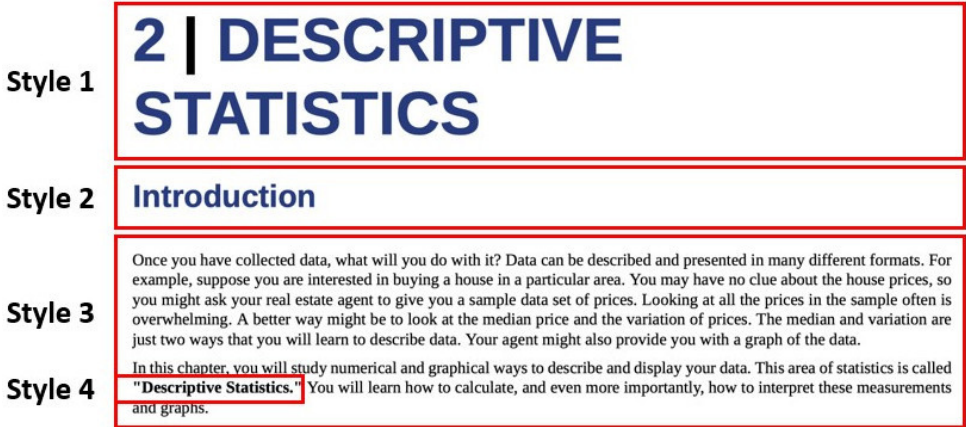


Figure 1.1: Example of hierarchical content and formatting styles. Content from Il-lowsky & Dean [135].

In this thesis, the content of textbooks is processed and represented hierarchically (words, words grouped into lines, lines grouped into fragments, fragments grouped into pages, and pages grouped into sections) in the extracted knowledge models.

1.2.2 Table of Contents

The Table of Contents (ToC) is a collection of references to the different structural elements that are designed to guide navigation and aid in learning. A ToC indicates not only the hierarchical arrangement of chapters and subchapters in the text but also provides an overview of the topics and subtopics of the textbooks. Due to this, a number of methods on ToC detection/analysis have been studied [68, 110] (👉 Chapter 2).

Independently from the textbook layout, each ToC reference provides the title of the chapter or subchapter, a start page number, and the relations with other sections (given by the hierarchy of sections). This universal property across books and domains makes the ToC the ideal starting point to analyze the structure of the textbooks. Other properties that facilitate the recognition of the ToC are [68]: contiguity (contiguous references to parts of the document), textual similarity (references share textual similarity with the referenced part), ordering (references appear in the same order as the parts of the document), optional elements (ToCs can include decorative elements), and no self-reference (all references are for other parts of the document). Figure 3.2.A (👉 page 57) shows an example of a ToC and its main elements, which provide information about the textbook's structure.

In this thesis, the ToC properties have been generalized into a set of rules that give insight into textbook structure and understanding. In the extracted knowledge models, each textbook section represents a structural component annotated with its textual content and relations to other sections. Additionally, each chapter/subchapter can potentially be treated as a topic/subtopic annotated with terms in the domain thanks to the explicit connections between the terms in the index section and the different content sections.

1.2.3 Index

A back-of-the-book index is a manually created and curated index of essential terms placed at the back of a textbook (also in nonfiction books or documents). The index not only lists the important terms and subjects of a textbook but also sorts them alphabetically, provides cross-references to and from related terms, and includes specific page numbers [263]. Since indexes are created by authors or dedicated human indexers¹⁰ following a predefined set of rules [17, 263], they can be viewed as reference models with meaningful links indicating where terms are introduced or elaborated in the textbook. Indices contain multiple components. Each index entry is identified by a main heading, normally a noun or noun phrase. A locator typically follows the heading. Locators are usually page numbers or ranges of page numbers. Index entries can have subentries, where the subheadings can form a grammatical relationship with the main heading. In this way, the structure of indices is usually

¹⁰For example, the American Society for Indexing has a indexer locator service at <https://www.asindexing.org/find-an-indexer/asi-indexer-locator/>

hierarchical. Finally, good indices contain cross-references among index terms. All these elements make the index a terminological network [21]. Figure 3.2.B (📖 page 57) shows an example of an index along with its multiple elements.

In this thesis, the guidelines specified by several textbook publishers to create indices have been generalized into a set of rules to extract domain terminology from the textbooks. Additionally, the index entries have been classified according to their domain specificity to discover concepts from the domain of the textbook and related domains. The extracted knowledge models encode the information about the index terms, concepts, and the relationships among them and the different sections of the textbooks. Such information makes the models machine-readable domain glossaries.

1.2.4 Other Structural and Formatting Elements

Textbooks contain multiple structural and formatting elements that provide semantics. Headings introduce chapters and subchapters. Page numbers identify each page and create connections with the ToC and the index terms. The formatting styles (font family, size, bold, and italic properties) identify different types or roles of text. Multiple columns, header/footer lines, and images are other elements that provide semantics to the textbooks. In Figure 1.1, the red squares identify four different formatting styles. Styles 1 and 2 are used for the main heading and sub-heading, respectively, and styles 3 and 4 characterize the body text and emphasized body text, respectively.

In this thesis, the structural and formatting elements are analyzed to properly recognize the content of the textbook. The generated knowledge models properly identify the headings for sections, page numbers, emphasized content, and the semantic roles of text.

1.3 Extraction of Knowledge Models

Textbook authors use their understanding of the domain when writing textbooks to explain domain knowledge to learners. Analyzing high-quality textbooks' formatting and structural elements makes it possible to automatically extract textbooks' encoded knowledge and gradually gain meaningful and valuable insights concerning a specific domain. This thesis presents a unified approach to extract, link, enrich, analyze, and formalize knowledge models from PDF textbooks. The extracted knowledge models are high-quality representations of the domain (Section 1.3.2).

1.3.1 Approach

Figure 1.2 presents the workflow for the extraction of knowledge models from textbooks. The approach has several inputs, outputs, phases, stages, and steps. Additionally, Figure 1.2 shows the connection between the different phases of the approach and the chapters of this thesis.

In the first phase of the approach (📖 Chapter 3), a textbook's structure, content, and domain terms are extracted. Structural information contains the list of chapters and subchapters of the textbook. The textbook's content is represented in a structured way (words, lines, text fragments, pages, and sections). Lastly, the domain

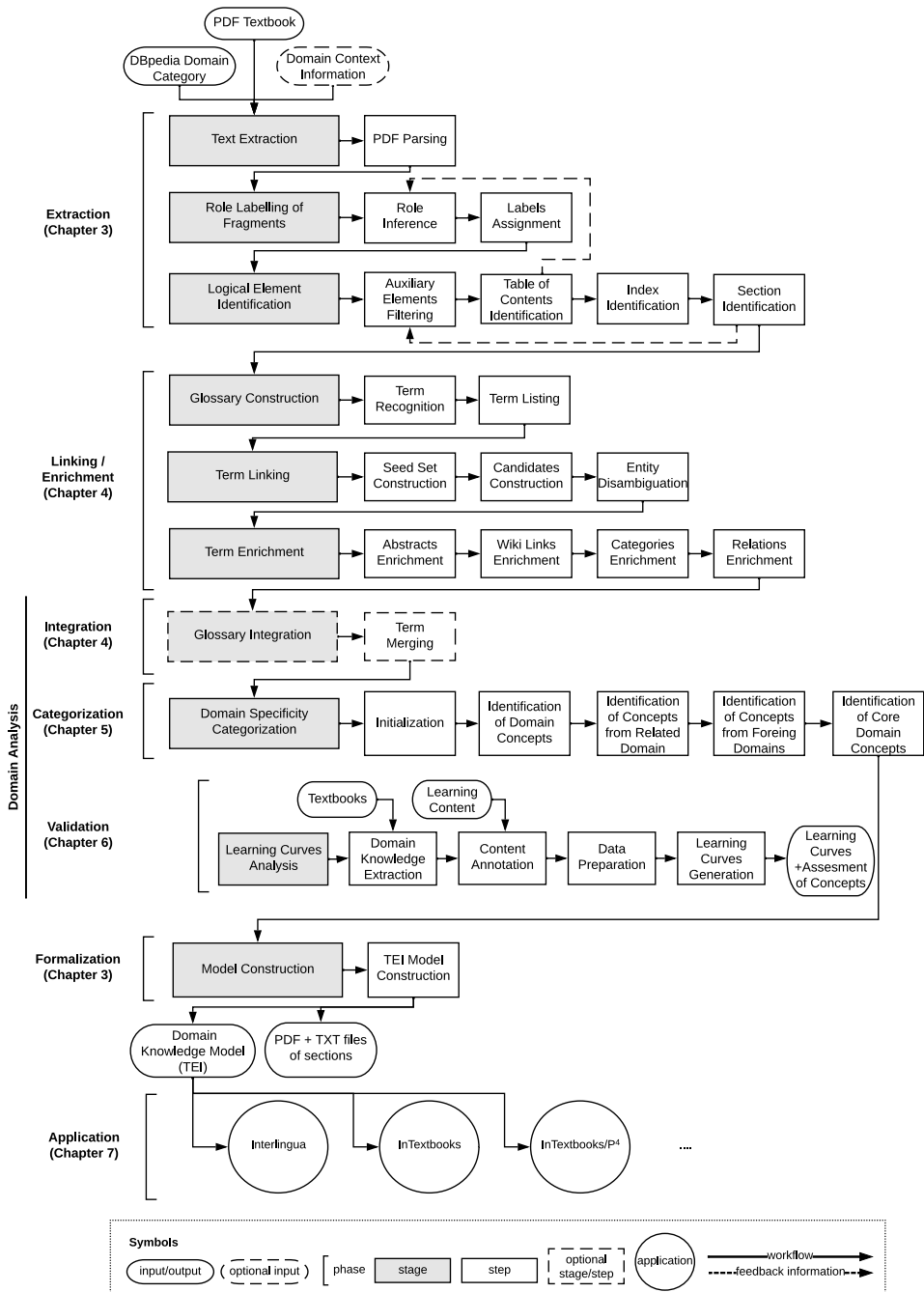


Figure 1.2: Unified approach for the extraction of knowledge models from textbooks and the chapters where each stage is discussed.

terms are extracted from the back-of-the-book index, which contains the terminology used in the textbook and the domain. In the next phase (🔗 Chapter 4), the domain terms are used as a bridge to link the textbooks to an external knowledge graph. Specifically, domain terms are matched to corresponding entities in DBpedia¹¹—a publicly available knowledge graph based on Wikipedia. The linking with DBpedia entities allows for the enrichment of the domain terms with semantic information (abstracts, Wikipedia links, categories, synonyms, and relations to other terms). In the third, fourth, and fifth phases, a domain analysis is performed. First, terms from multiple textbooks are integrated into a single model to get better coverage of the target (or main) domain (🔗 Chapter 4). Then, terms are categorized according to their relevance to the target domain to identify the concepts that belong to the textbook’s main domain, related domains, or unrelated domains (🔗 Chapter 5). After that, the validity of the extracted concepts as knowledge components are established using learning curve analysis (🔗 Chapter 6). The final phase of the automatic approach is formalization, where all the extracted knowledge is serialized as a descriptive XML file using the Text Encoding Initiative (🔗 Chapter 3, Section 3.3.5). After the approach has produced the knowledge models, these are ready to be used in various applications (🔗 Chapter 7).

The proposed approach does not aim to extract formal ontologies but domain knowledge models, where elementary knowledge elements or concepts form a knowledge space, as used in traditional adaptive educational hypermedia systems [38]. This kind of model is less formal than Descriptive Logic ontologies, but it supports a wide range of educational applications by focusing on meaningful information (concepts).

In summary, this thesis’ unified approach extracts an initial set of information (structure, content, and terms) from the textbooks, then gradually adds new information (links and semantic content), and finally analyses and refines the knowledge about the domain (concepts). The final results are domain-oriented knowledge models extracted from textbooks.

1.3.2 Quality

The extracted knowledge models are not guaranteed to provide high-quality domain representations. They can suffer from potential problems such as low coverage, poor granularity, and lack of semantics. Different quality properties need to be defined to evaluate the models. For example, ontologies—as reusable representations of knowledge—have been evaluated using accurate, well-defined, and easy-to-apply metrics [188]. Gómez-Pérez [113] and Gruber [115] defined important qualities an ontology must possess. Similarly, other authors [47, 143, 186, 259] have defined their quality factors.

Based on the different quality metrics proposed for ontologies, the following properties are defined to verify the quality of the extracted models:

1. Accuracy: The textbook’s information is correctly represented in the knowledge models. Similar to the *accuracy* property defined by Burton-Jones et al. [47], and the *class precision* metric defined by Mc Gurk et al. [186].

¹¹<https://www.dbpedia.org/>

2. **Semantics:** The knowledge models contain additional meaningful and truthful data from an external knowledge base. Similar to the *consistency* property defined by Gómez-Pérez [113] and Tankeleviciene & Damasevicius [259] in the sense that terms linked to external entities provide unambiguous information.
3. **Coverage:** The knowledge models include a significant area of the target domain. Similar to the *completeness* property defined by Gómez-Pérez [113] and Tankeleviciene & Damasevicius [259], and the *class coverage* metric defined by Mc Gurk et al. [186].
4. **Specificity:** The concepts in the knowledge models have an identified relevance to the target domain. Similar to the *specificity* property defined by Tankeleviciene & Damasevicius [259].
5. **Cognitive validity:** The domain concepts in the knowledge models are valid knowledge components for knowledge modeling in the target domain. Similar to the evaluation of the *domain model's knowledge components* by Martin et al. [182].
6. **Granularity:** The domain concepts in the knowledge models are fine-grained knowledge components in the target domain. Similar to the evaluation of the *domain model's granularity* by Martin et al. [182].

The sections where each of the properties are discussed in this thesis are shown in Figure 1.3.

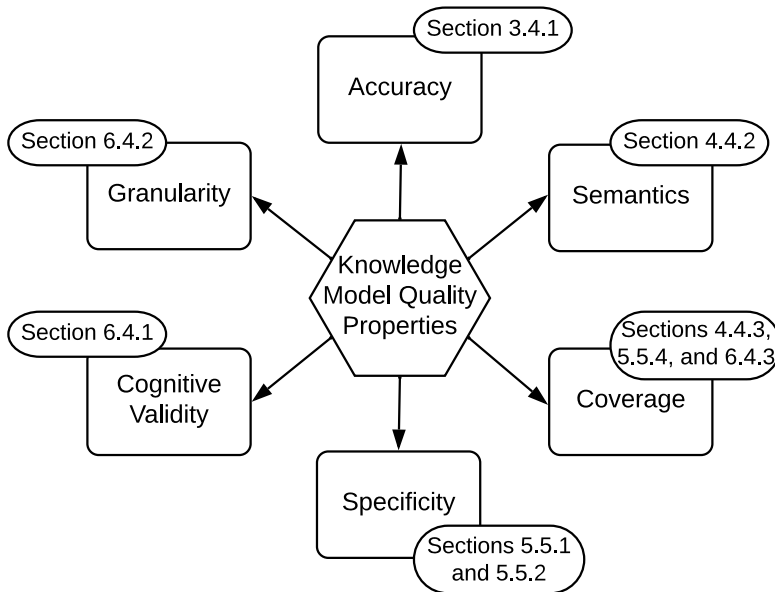


Figure 1.3: Quality properties of the extracted knowledge models and the sections where they are discussed.

1.4 Research Questions

The main research question of this thesis is:

RQMain Can high-quality and domain-specific knowledge models be automatically extracted from textbooks?

Multiple subquestions are defined to help finding an answer to the main research question. These subquestions incorporate the quality properties (shown in *italics*) defined previously. The subquestions are:

- RQ1** What are the characteristics of existing approaches to extract information elements from textbooks? (Explored in Chapter 2).
- RQ2** Can the structure, content, and domain terms be automatically extracted from textbooks, and if so, what is the *accuracy* and value of the extracted information? (Explored in Chapter 3).
- RQ3** Can the domain terms extracted from textbooks be linked to their corresponding entities in a global reference model, and if so, what are the *semantics* obtained from the linkage? (Explored in Chapter 4).
- RQ4** Can the domain terms extracted from multiple textbooks be integrated into a single model, and if so, what is the *coverage* of such an integrated model? (Explored in Chapters 4, 5, and 6).
- RQ5** Can the domain relevance of concepts extracted from textbooks be established, and if so, what is the *specificity* of such concepts? (Explored in Chapter 5).
- RQ6** Are the concepts extracted from textbooks *cognitively valid* components for knowledge modeling, and if so, what is the *granularity* of such concepts? (Explored in Chapter 6).
- RQ7** Is knowledge extraction effective across multiple domains? (Explored in Chapters 3, 4, 5, and 6).
- RQ8** How can the knowledge models extracted from textbooks support AIES? (Explored in Chapter 7).

These research questions will be revisited and answered in the concluding chapter of this thesis (👉 Chapter 8).

1.5 Thesis Structure

The thesis comprises six main chapters, each describing a part of the extraction of knowledge models (and their respective evaluation) or the application of such models. Four (Chapters 3, 4, 5, and 7) of the primary chapters are modified versions of published papers. The remaining two (Chapters 2 and 6) contain new research results. Chapter 2 is unpublished and Chapter 6 has been submitted for publication.

Chapter 2 — Knowledge Extraction from Unstructured and Semi-Structured Textual Resources

This chapter is the unpublished related-work section of this thesis. In this chapter, different approaches for knowledge extraction from unstructured and semi-structured textual resources are described and discussed.

Chapter 3 — Order out of Chaos: Construction of Knowledge Models from PDF Textbooks

In this chapter, the extraction and formalization phases of the approach are described. This chapter is a modified version of the paper:

Alpizar-Chacon, I. & Sosnovsky, S., “Order out of chaos: construction of knowledge models from pdf textbooks”, in: *Proceedings of the ACM Symposium on Document Engineering 2020*, 2020, pp. 1–10

The changes come from an updated version of this paper that has been published in a journal:

Alpizar-Chacon, I. & Sosnovsky, S., “Knowledge models from pdf textbooks”, *New Review of Hypermedia and Multimedia*, vol. 27, no. 1-2, 2021, pp. 128–176

My contribution included developing the software, defining the rules for the extraction of the knowledge models, performing the evaluation, and writing most of the paper.

Chapter 4 — Expanding the Web of Knowledge: One Textbook at a Time

In this chapter, the linking/enrichment and integration phases of the approach are described. This chapter is a modified version of the paper:

Alpizar-Chacon, I. & Sosnovsky, S., “Expanding the web of knowledge: one textbook at a time”, in: *Proceedings of the 30th on Hypertext and Social Media*, HT ’19, Hof, Germany: ACM, 2019 (ACM HT 19 Ted Nelson Newcomer Award)

The changes also come from the journal publication:

Alpizar-Chacon, I. & Sosnovsky, S., “Knowledge models from pdf textbooks”, *New Review of Hypermedia and Multimedia*, vol. 27, no. 1-2, 2021, pp. 128–176

My contribution included developing the software, defining the rules for the linking of index terms to DBpedia entities, performing the evaluation, and writing most of the paper.

Chapter 5 — What’s in an Index: Extracting Domain-specific Knowledge Graphs from Textbooks

In this chapter, the categorization phase of the approach is described. This chapter is a modified version of the paper:

Alpizar-Chacon, I. & Sosnovsky, S., “What’s in an index: extracting domain-specific knowledge graphs from textbooks”, in: *Proceedings of the ACM Web Conference 2022 (WWW ’22)*, 2022, pp. 966–976

To include it in this thesis, minor changes, in particular removing repetitive content, were made to the original paper. My contribution included developing the software, defining the algorithms for the categorization of the index terms according to their domain specificity, performing the evaluation, and writing most of the paper.

Chapter 6 — The Power of the Curve: Measuring the Quality of Extracted Concepts Using Learning Curve Analysis

This chapter is new work. In this chapter, learning curve analysis is used in an experiment to see if the extracted concepts are valid units of knowledge modeling. A modified version of this chapter has been submitted for publication:

Alpizar-Chacon, I. & Sosnovsky, S., “Measuring the quality of domain models extracted from textbooks with learning curves analysis”, *In submission*, n.d.

Chapter 7 — Lights, Camera, Action! Applications of Textbook Knowledge Models

This chapter describes three AIES where the extracted knowledge models are used, and is based on three papers:

Alpizar-Chacon, I. & Sosnovsky, S., “Interlingua: linking textbooks across different languages”, in: *Proceedings of the First Workshop on Intelligent Textbooks*, vol. 2384, CEUR-WS, 2019, pp. 104–117

My contribution included adapting and improving the software, describing the process, and writing most of the paper.

Alpizar-Chacon, I., Hart, M. van der, Wiersma, Z. S., Theunissen, L. S. & Sosnovsky, S., “Transformation of pdf textbooks into intelligent educational resources”, in: *Proceedings of the Second Workshop on Intelligent Textbooks*, vol. 2674, CEUR-WS, 2020, pp. 4–16

My contribution included guiding the definition of the system’s architecture, guiding the system’s development, describing the process, and writing most of the paper.

Alpizar-Chacon, I., Barria-Pineda, J., Akhuseyinoglu, K., Sosnovsky, S. & Brusilovsky, P., “Integrating textbooks with smart interactive content for learning programming”, in: *Proceedings of the Third Workshop on Intelligent Textbooks*, vol. 2895, CEUR WS, 2021, pp. 4–18

My contribution included participating in the discussions, integrating content with the textbooks, and participating in writing the paper.

Chapter 8 — Conclusion

This last chapter provides a conclusion and reflection on the topic of this thesis: automatic extraction of knowledge models from textbooks. The research questions are answered and discussed. Limitations and future work are discussed as well.

Other Work

The following papers are relevant work where the generated knowledge models have been used, but are not a part of this thesis:

Dresscher, L., Alpizar-Chacon, I. & Sosnovsky, S., “Generation of assessment questions from textbooks enriched with knowledge models”, in: *Proceedings of the Third Workshop on Intelligent Textbooks*, vol. 2895, CEUR-WS, 2021, pp. 45–59

Javadian Sabet, A., Alpizar-Chacon, I., Barria-Pineda, J., Brusilovsky, P. & Sosnovsky, S., “Enriching intelligent textbooks with interactivity: when smart content allocation goes wrong”, in: *Proceedings of the Fourth Workshop on Intelligent Textbooks*, vol. 3192, CEUR-WS, 2022, pp. 92–104

The <https://intextbooks.science.uu.nl/> site can be consulted for an up-to-date list of the research associated with this thesis.

Knowledge Extraction from Unstructured and Semi-Structured Textual Resources

Abstract In this thesis, knowledge extraction from digital textbooks is explored. Textbooks are high-quality and formatted documents, but retrieving their structure from PDF resources is challenging. Therefore, from the perspective of knowledge extraction, PDF textbooks are considered to be unstructured. This chapter analyzes different approaches for knowledge extraction from unstructured and semi-structured textual resources. The studied approaches are described in terms of seven features related to textbook components. Furthermore, the main findings across all features are reported and discussed. The analysis showed that no approach extracts and represents information across content, structure, and domain textbook elements.

2.1 Introduction

Information extraction dates back to the '60s and '70s, but it grew very rapidly from the late '80s [106]. In the '90s, new computational methods were needed to support the extraction of valuable knowledge from the rapidly growing volumes of data [95]. In particular, the discovery of knowledge in texts has been widely studied [92, 156, 225]. Knowledge extraction starts by first recognizing the content of the documents. One of the most traditional and established techniques to extract the content from textual documents is Optical Character Recognition (OCR). OCR deals with the problem of recognizing optically processed characters [86]. OCR techniques and devices have been available since the middle of the 1950s, and their development was motivated by the need to handle the enormous amount of paper documents [114]. However, there has been a shift from paper to digital, in which PDF documents are extensively adopted. According to the CommonCrawl database¹, PDF is the second most popular file format on the web (after HTML). The use of digital formats, such as PDF, has motivated a new set of techniques to handle textual documents. The approach by Lovegrove & Brailsford [173] has been recognized as the first one that deals with the analysis of PDF files [127]. Modern techniques usually focus on recognizing specific objects of digital documents, such as figures [59].

After content recognition, a process to recognize the structure of the documents is necessary to achieve knowledge extraction [213]. For example, Zhou et al. [301] implemented a system to recognize semi-structured patient records and extract three types of information (numeric values, medical terms, and categorical values). In a more recent approach, Kruit et al. [159] described a 3-step pipeline to build a Knowledge Base (KB) from tables in scientific publications. Due to the diversity of knowledge extraction approaches and their scopes, they can be compared along several axes: the types of textual resources used as input, the different applied methods or techniques, the specific objects that are recognized and extracted, the enrichment of the gathered knowledge, the used knowledge representation mechanisms, among others.

In this thesis, the main object of interest is digital (PDF) textbooks and the knowledge that can be extracted from them. From the perspective of knowledge extraction, textbooks are considered unstructured resources due to the difficulty of extracting their semantic structure. This chapter presents and analyzes 46 approaches for knowledge extraction from unstructured and semi-structured textual resources that can be used to extract knowledge from textbooks. Seven features have been used for the analysis. One of them is used to compare the approaches in terms of the extracted components. These components are defined according to the knowledge extracted from textbooks. The main contribution of this analysis is the overview of the different approaches and challenges concerning knowledge extraction from textbooks and other textual documents.

This chapter is organized as follows. Section 2.2 presents relevant concepts and defines the scope of the performed analysis. In Section 2.3, the used features are

¹<https://commoncrawl.github.io/cc-crawl-statistics/plots/mimetypes>

described, and the studied approaches are discussed. Section 2.4 presents an additional discussion. Limitations are discussed in Section 2.5. Finally, Section 2.6 concludes the chapter.

2.2 Preliminaries

This section introduces the concepts relevant to this chapter: knowledge, knowledge extraction, and the different kinds of information resources. Additionally, the scope of this analysis is stated.

2.2.1 Knowledge

The Merriam-Webster dictionary defines knowledge as "the fact or condition of knowing something with familiarity gained through experience or association" and "the sum of what is known" [194]. In the information and knowledge literature, the Data-Information-Knowledge-Wisdom (DIKW) hierarchy is often used to define data, information, and knowledge in terms of one another [234]. For example, Turban et al. define knowledge as "data and/or information that have been organized and processed to convey understanding, experience, accumulated learning, and expertise as they apply to a current problem or activity" [271]. As another example, Medford et al. use the DIKW hierarchy to define knowledge in the context of heterogeneous catalysis [190]. Following the same direction, knowledge is defined in this thesis as *the integration of information or data to gain meaningful and valuable insights in a specific domain*. This definition guides the selection and analysis of related work in this chapter and functions as a common element between the chapters of this thesis.

Traditionally, knowledge is obtained by humans using different strategies [160]. In this chapter, knowledge acquisition performed by computers is explored. Knowledge extraction is the process of acquiring hidden knowledge from the different types of information using a computational approach. This process involves different techniques that combine the expertise of humans and machines [221].

The extracted knowledge should be represented in a structured way. Yao et al. argue that knowledge is built on concepts and the relations among them. Additionally, they claim that a knowledge structure is built based on hierarchical structures of concepts. In this context, a concept is interpreted as a granule or a basic unit. It is also possible to decompose a concept into a family of smaller units. Finally, Yao et al. mention that tree structures, concept graphs, and semantic networks are knowledge structures [295].

2.2.2 Kinds of Information Resources

There are three kinds of information resources from the perspective of knowledge extraction: unstructured, semi-structured, and structured.

Unstructured

In a classical definition, unstructured information is data that cannot be stored in rows and columns in a relational database [32]. In a more general description,

unstructured information means that no identifiable semantic structure is available [248]. Unstructured information resources are typically text-heavy, including word processing documents, PDFs, and social media/messaging system content [236]. Unstructured textual information is also called free text [119, 250].

Semi-Structured

Semi-structured information refers to self-describing terms where there is no separate description of the type or structure of the data [1]. This kind of information does not require a schema definition [248]. Semi-structured information contains tags or elements to separate the semantics from the content and enforce hierarchies of records and fields within the data [236]. For example, XML and CSV documents are semi-structured resources [187].

Structured

Structured information conforms to a standard schema or type [20]. The typical example of fully structured information is a relational database system, where the schema has to be defined before the database is populated with the content [248].

2.2.3 Scope

As mentioned before, knowledge extraction from digital textbooks is the main interest of this thesis. Therefore, this chapter analyzes knowledge extraction approaches for digital textbooks, particularly in PDF format. This analysis also considers other unstructured or semi-structured textual information resources that share elements or features with digital textbooks. For example, approaches that recognize the structure of titles and subtitles in scientific articles [26] or generate keyphrases for any text [296] are also considered. The evaluation of the different approaches is done using seven features described in the following section (🔗 Section 2.3.1).

Once knowledge has been extracted from textbooks, it can be used in a wide range of intelligent services. For example, in social navigation and annotation [25], adaptation [254], matching [185], and linking [193]. The application of the knowledge extracted from digital textbooks (and other textual information resources) is beyond the scope of this chapter. Also, note that, the analysis presented in this chapter is not an exhaustive research of all the techniques and kind of resources that have been studied in the field of knowledge extraction. The purpose of this analysis is to provide a representative overview of the approaches that are most relevant to the research presented in this thesis.

2.3 Approaches for Knowledge Extraction from Textual Resources

This section first defines the features used to analyze all the approaches from a perspective of knowledge extraction from digital textbooks. Then, an overview of the studied approaches is provided. Finally, there is a discussion of the main findings.

2.3.1 Features

In this analysis, the definitions of knowledge from Section 2.2.1 and knowledge structures from Yao et al. [295] are used to portray 11 elements (or concepts) representing units of information that can be extracted from textbooks. The elements are divided into content, structure, and domain. Additionally, the three categories form a combined feature to analyze the approaches. Furthermore, when elements from the three categories are aggregated in a machine-readable format, the resulting knowledge structure is referred to as a Textbook Knowledge Model (TKM) in this chapter. Finally, six more features that describe the approaches in a general way are also used.

Textbook elements feature

This combined feature describes the 11 textbook elements extracted by the approaches. Additionally, different annotations indicate more specific elements or characteristics of the approaches. The following lists mention which data from the textual resources are recognized and extracted as part of the main element.

Content elements:

- *Layout*: characters' bounding boxes, coordinates, and rotation angles. Also, font characteristics, images, and vector elements.
- *Content (text)*: words, lines, text fragments, and pages for each section. The *partial* annotation indicates that the approach does not extract the content for the whole document, but only for a specific aspect of it (e.g., headers or tables).
- *Content (objects)*: special content objects. The following annotations are used to indicate the extracted objects: *geometrical figures*, *images*, *tables*, *formulae*, *pseudo-codes*, and *algorithmic procedures*.
- *Header metadata*: general header metadata (e.g., title, authors, and affiliation).
- *Citation metadata*: citation and bibliographic metadata.

Structure elements:

- *Organization (ToC)*: the Table of Contents. The *section headers* annotation indicates that only the section headers are recognized without creating an organizational structure.

Domain elements:

- *Terms*: the important terms or keyphrases used in a document. The following annotations are used to indicate the source of the extracted terms: *index* (from the back-of-the-book index), *content* (from the textual content), *Wikipedia titles* (from Wikipedia's articles), and *generated* (from a generative model). The *indirectly* annotation indicates that terms can be identified from other elements, e.g., named entities.
- *Named Entities*: Named Entities linked to the extracted terms. For this element, a Named Entity Disambiguation process is required to resolve potential ambiguity among several possible candidates [51].

- **Concepts:** terms identified as relevant in a target domain. The following annotations are used to indicate the level of relevance concerning the domain of interest: *core* (the most important concepts), *main* (other concepts in the domain), *related* (concepts from related domains), *unrelated* (concepts not related to the main domain), and *list* (concepts belonging to one domain from a closed list). The *indirectly* annotation indicates that the domain concepts can be extracted indirectly.
- **Topics:** topics in the textual resources. Annotations indicate the exact method that is used to extract the topics.
- **Relations:** pedagogical relations among concepts. There is an emphasis on prerequisite-outcome relations. Annotations indicate the exact type of identified relations.

General features

The general features are:

- **Year:** year of publication.
- **Input:** textual resources for which the approach was developed. The possible values are: textbooks, books, scientific articles, software-requirement documents, Massive Open Online Courses (MOOCs), and general documents.
- **Methods:** the different types of methods used in the approach. The values are: rule-based (rules or heuristics), text-based (classical information retrieval techniques), NLP-based (Natural Language Processing techniques and models), semantic-based (semantic similarity measures), learning-based (supervised or unsupervised machine learning), BERT-based (BERT based model), graph-based (graph representations and algorithms), knowledge-based (use of KBs), methodology-based (definition of a specific methodology), manual (manual process or methodology), and semi-automatic (method includes human supervision). It is important to note that these terms are not disjoint (e.g., BERT-based models are part of NLP), but this classification was chosen to inform about the more concrete methods used in the approaches.
- **External KB:** any external Knowledge Base used or required in the approach.
- **Output:** the format in which the extracted elements are represented. *Ontology* refers to languages used for defining ontologies (e.g., RDF and OWL). A hyphen (-) is used when it is impossible to infer the output format from the approach's publication.
- **Knowledge Model:** if the approach produces a TKM.

2.3.2 Approaches

Forty-six approaches (corresponding to 51 publications) were analyzed in this study. The publications were selected using the following method. The starting point was a collection of 25 papers already identified in the initial stages of the research presented in this thesis. Using those papers, the list of 11 textbook elements that can be extracted from textbooks was defined. Afterward, Google Scholar was used to

Table 2.1: Overview of the content, structure, and domain elements extracted in the analyzed approaches.

Approach	Content					Structure	Domain				
	Layout	Content (text)	Content (objects)	Header metadata	Citation metadata	Organization (ToC)	Terms	Named Entities	Concepts	Topics	Relations
Chao & Fan [52]	x	x	x								
Shao & Futrelle [247]		x	x								
Hassan [127]	x	x	x								
Baker et al. [23]		x	x								
Oro & Ruffolo [215]	x	x	x								
Lin et al. [168]	x	x	x								
Kern et al. [151] / Kern & Klampfl [152]	x	x		x	x						
Fang et al. [93]	x	x	x								
Kruit et al. [159]			x					x			
Gao et al. [110]	x	x				x					
Gao et al. [108]	x	x	x	x		x					
Ramanathan et al. [226]	x	x				x					
Wu et al. [293]	x	x				x					
Tkaczyk et al. [265]	x	x		x	x	x					
Wu et al. [290]		x	x	x	x	x					
Bast & Korzen [26]	x	x		x		x					
Tuorob et al. [269, 270]	x	x	x			x					
Larrañaga et al. [165]							x				x
Wali et al. [280]							x				
Lopes et al. [171, 172]							x				
Dwarakanath et al. [82]							x				
Wang et al. [284, 285]							x				x
Mihalcea & Csomai [195]							x	x			
Medelyan et al. [189]							x			x	
Milne & Witten [197]							x	x			
Mendes et al. [192] / Daiber et al. [63]							x	x			
Moro et al. [204]							x	x			
Zhu & Iglesias [302]							x	x			
Aghaebrahmanian & Cielieback [5]							x	x			
Chaudhri et al. [55]							x				
Yuan et al. [296]							x				
Thaker et al. [260]									x		
Wang et al. [283]									x		
Chau et al. [54]									x		
Xu et al. [294]							x		x		
Rigutini et al. [229]									x		
Kida et al. [153]									x		
Gaizauskas et al. [105]									x		
Chaplot et al. [53]									x		x
Labutov et al. [161]											x
Adorni et al. [4]											x
Guerra et al. [117]										x	
Kawamata et al. [147]									x		
Larrañaga et al. [164]		x				x	x				x
Fiallos & Ochoa [99]	x	x				x	x			x	
Thareja et al. [262]	x	x				x	x				x

find more relevant papers for each element. Only the most representative papers for each of the 11 elements were selected during the search.

Table 2.1 presents an overview of the elements extracted in the 46 analyzed approaches. The approaches are referred by the publication(s) where they are described. Approaches are ordered using the three element categories. After indexing (coding) the papers, four clusters have emerged:

- **C1:** approaches that only extract content elements. There are nine approaches in this cluster, which is identified with the color. The approach by Kruit et al. [159] is a particular case. Even though they perform entity linking, the recognized entities are local from the extracted table data. Therefore, the approach is placed in this cluster.
- **C2:** approaches that extract both content and structure elements. There are eight approaches in this cluster, which is identified with the color.

Table 2.2: Detailed information about the analyzed papers (1/11).

	Chao & Fan [52]	Shao & Futrelle [247]	Hassan [127]	Baker et al. [23]
Cluster	C1	C1	C1	C1
Year	2004	2005	2009	2009
Input	general documents	general documents	general documents	general documents
Methods	rule-based	rule-based, learning-based	rule-based	rule-based
External KB				
<i>Layout</i>	x		x	
<i>Content (text)</i>	x	x (partial)	x	x (partial)
<i>Content (objects)</i>	x (images)	x (geometrical figures)	x (tables, images)	x (formulae)
<i>Header metadata</i>				
<i>Citation metadata</i>				
<i>Organization (ToC)</i>				
<i>Terms</i>				
<i>Named Entities</i>				
<i>Concepts</i>				
<i>Topics</i>				
<i>Relations</i>				
Output	XML	-	XML	LaTeX, MathML
Knowledge model	no	no	no	no

- **C3:** approaches focused on domain elements. There are 27 approaches in this cluster, which is identified with the C3 color. Approaches in this cluster either get the required content by manual pre-processing (e.g., Larrañaga et al. [165]) or as an input in their process (e.g., Lopes et al. [171]); therefore, they do not extract content items directly.
- **C4:** approaches that extract knowledge across all three categories. There are three approaches in this cluster, which is identified with the C4 color.

In general, for each cluster, approaches are ordered by publication year. However, related approaches by the same authors or research groups are placed together.

Tables 2.2-2.12 provide detailed information using the seven defined features and their respective annotations.

2.3.3 Main Findings

The defined features are used in this subsection to discuss and analyze the approaches. The six general features group the findings. The textbook elements are used across the general features to guide the discussion. For each point of discussion, example approaches are cited. These mentions are not exhaustive; they are used to support the arguments. The reader can use Tables 2.1-2.12 to focus on the approaches with a specific set of characteristics.

Year

Figure 2.1 presents the distribution of approaches by year of publication and cluster. In general, the extraction of knowledge from textual resources has stayed active since the 2000s.

C1 approaches are the point of departure for knowledge extraction since they extract the content of the resources directly, and therefore, they are the oldest in the analyzed set. Initial approaches for textual extraction of PDF documents belong to

Table 2.3: Detailed information about the analyzed papers (2/11).

	Oro & Ruffolo [215]	Lin et al. [168]	Kern et al. [151] / Kern & Klampfl [152]	Fang et al. [93]	Kruit et al. [159]
Cluster	C1	C1	C1	C1	C1
Year	2009	2014	2013	2011	2020
Input	general documents	general documents	scientific articles	general documents	scientific articles
Methods	rule-based	rule-based, learning-based	rule-based, learning-based	rule-based	rule-based, learning-based
External KB					Semantic Scholar
<i>Layout</i>	x	x	x	x	
<i>Content (text)</i>	x (partial)	x (partial)	x	x (partial)	
<i>Content (objects)</i>	x (tables)	x (formulae)		x (tables)	x (tables)
<i>Header metadata</i>			x		
<i>Citation metadata</i>			x		
<i>Organization (ToC)</i>					
<i>Terms</i>					
<i>Named Entities</i>					x (local entities)
<i>Concepts</i>					
<i>Topics</i>					
<i>Relations</i>					
Output	XML	-	-	-	ontology
Knowledge model	no	no	no	no	no

Table 2.4: Detailed information about the analyzed papers (3/11).

	Gao et al. [110]	Gao et al. [108]	Ramanathan et al. [226]	Wu et al. [293]
Cluster	C2	C2	C2	C2
Year	2009	2011	2012	2013
Input	books	books	books	books
Methods	rule-based, learning-based	rule-based, learning-based	rule-based	rule-based
External KB				
<i>Layout</i>	x	x	x	x
<i>Content (text)</i>	x (partial)	x (partial)	x (partial)	x (partial)
<i>Content (objects)</i>		x (images)		
<i>Header metadata</i>		x		
<i>Citation metadata</i>				
<i>Organization (ToC)</i>	x	x	x	x
<i>Terms</i>				
<i>Named Entities</i>				
<i>Concepts</i>				
<i>Topics</i>				
<i>Relations</i>				
Output	-	XML	PDF with bookmarks	-
Knowledge model	no	no	no	no

Table 2.5: Detailed information about the analyzed papers (4/11).

	Tkaczyk et al. [265]	Wu et al. [290]	Bast & Korzen [26]	Tuarob et al. [269, 270]
Cluster	C2	C2	C2	C2
Year	2015	2015	2017	2016
Input	scientific articles	scientific articles	scientific articles	scientific articles
Methods	learning-based	rule-based, learning-based	rule-based	rule-based, learning-based
External KB				
<i>Layout</i>	x		x	x
<i>Content (text)</i>	x	x	x	x (partial)
<i>Content (objects)</i>		x (tables, images)		x (pseudo-codes, algorithmic procedures)
<i>Header metadata</i>	x	x	x	
<i>Citation metadata</i>	x	x		
<i>Organization (ToC)</i>	x (section headers)	x (section headers)	x (section headers)	x
<i>Terms</i>				
<i>Named Entities</i>				
<i>Concepts</i>				
<i>Topics</i>				
<i>Relations</i>				
Output	NLM (XML)	-	text/XML/JSON	text
Knowledge model	no	no	no	no

Table 2.6: Detailed information about the analyzed papers (5/11).

	Larrañaga et al. [165]	Wali et al. [280]	Lopes et al. [171, 172]	Dwarakanath et al. [82]
Cluster	C3	C3	C3	C3
Year	2004	2013	2010	2013
Input	textbooks	textbooks	general documents	software-requirements documents
Methods	rule-based, NLP-based, semi-automatic	rule-based, knowledge-based	rule-based, NLP-based	NLP-based
External KB		Wikipedia		
<i>Layout</i>				
<i>Content (text)</i>				
<i>Content (objects)</i>				
<i>Header metadata</i>				
<i>Citation metadata</i>				
<i>Organization (ToC)</i>				
<i>Terms</i>	x (index)	x (index)	x (content)	x (content)
<i>Named Entities</i>				
<i>Concepts</i>				
<i>Topics</i>				
<i>Relations</i>	x (part-of, is-a, prerequisite, and next)			
Output	ontology	XML	text	text
Knowledge model	no	no	no	no

Table 2.7: Detailed information about the analyzed papers (6/11).

	Wang et al. [284, 285]	Mihalcea & Csomai [195]	Medelyan et al. [189]	Milne & Witten [197]
Cluster	C3	C3	C3	C3
Year	2016	2007	2008	2008
Input	textbooks	general documents	general documents	general documents
Methods	rule-based, semantic-based, learning-based, knowledge-based	text-based, learning-based, knowledge-based	semantic-based, learning-based, knowledge-based	semantic-based, learning-based, knowledge-based
External KB	Wikipedia	Wikipedia	Wikipedia	Wikipedia
<i>Layout</i>				
<i>Content (text)</i>				
<i>Content (objects)</i>				
<i>Header metadata</i>				
<i>Citation metadata</i>				
<i>Organization (ToC)</i>				
<i>Terms</i>	x (Wikipedia titles)	x (Wikipedia titles)	x (Wikipedia titles)	x (indirectly)
<i>Named Entities</i>		x		x
<i>Concepts</i>				
<i>Topics</i>			x (aggregation of terms)	
<i>Relations</i>	x (prerequisite)			
Output	graph	HTML	-	-
Knowledge model	no	no	no	no

Table 2.8: Detailed information about the analyzed papers (7/11).

	Mendes et al. [192] / Daiber et al. [63]	Moro et al. [204]	Zhu & Iglesias [302]	Aghaebrahimian & Cieliebak [5]
Cluster	C3	C3	C3	C3
Year	2013	2014	2018	2020
Input	general documents	general documents	general documents	general documents
Methods	NLP-based, semantic-based, knowledge-based	semantic-based, graph-based, knowledge-based,	NLP-based, semantic-based, learning-based, knowledge-based	semantic-based, learning-based
External KB	DBpedia	Babel-Net	DBpedia	proprietary knowledge base
<i>Layout</i>				
<i>Content (text)</i>				
<i>Content (objects)</i>				
<i>Header metadata</i>				
<i>Citation metadata</i>				
<i>Organization (ToC)</i>				
<i>Terms</i>	x (indirectly)	x (indirectly)	x (indirectly)	x (indirectly)
<i>Named Entities</i>	x	x	x	x
<i>Concepts</i>				
<i>Topics</i>				
<i>Relations</i>				
Output	JSON/HTML/XML/RDF	-	-	-
Knowledge model	no	no	no	no

Table 2.9: Detailed information about the analyzed papers (8/11).

	Chaudhri et al. [55]	Yuan et al. [296]	Thaker et al. [260]	Wang et al. [283]
Cluster	C3	C3	C3	C3
Year	2021	2018	2019	2020
Input	textbooks	general documents	textbooks	textbooks
Methods	learning-based, BERT-based	learning-based	text-based, NLP-based, knowledge-based Wikipedia	methodology-based, manual
External KB				
<i>Layout</i>				
<i>Content (text)</i>				
<i>Content (objects)</i>				
<i>Header metadata</i>				
<i>Citation metadata</i>				
<i>Organization (ToC)</i>				
<i>Terms</i>	x (content)	x (generated)		
<i>Named Entities</i>				
<i>Concepts</i>			x (main)	x (main and related)
<i>Topics</i>				
<i>Relations</i>				
Output	graph	text	text	text
Knowledge model	no	no	no	no

Table 2.10: Detailed information about the analyzed papers (9/11).

	Chau et al. [54]	Xu et al. [294]	Rigutini et al. [229]	Kida et al. [153]	Gaizauskas et al. [105]
Cluster	C3	C3	C3	C3	C3
Year	2021	2002	2006	2007	2014
Input	textbooks	general documents	general documents	general documents	general documents
Methods	NLP-based, learning-based, knowledge-based	Text-based, NLP-based, knowledge-based	learning-based	rule-based, semantic-based	semantic-based
External KB	Wikipedia	GermaNet, WordNet	WWW (using a search engine)	WWW (using a search engine)	Wikipedia, EuroVoc
<i>Layout</i>					
<i>Content (text)</i>					
<i>Content (objects)</i>					
<i>Header metadata</i>					
<i>Citation metadata</i>					
<i>Organization (ToC)</i>					
<i>Terms</i>		x (content)			
<i>Named Entities</i>					
<i>Concepts</i>	x (main and related)	x (list)	x (list)	x (main, related, and unrelated)	x (list)
<i>Topics</i>					
<i>Relations</i>					
Output	text	-	-	-	-
Knowledge model	no	no	no	no	no

Table 2.11: Detailed information about the analyzed papers (10/11).

	Chaplot et al. [53]	Labutov et al. [161]	Adorni et al. [4]	Guerra et al. [117]	Kawamata et al. [147]
Cluster	C3	C3	C3	C3	C3
Year	2016	2017	2019	2013	2021
Input	MOOCs	textbooks	textbooks	textbooks	textbooks
Methods	text-based, NLP-based	learning-based	rule-based	NLP-based	NLP-based
External KB					Wikipedia
<i>Layout</i>					
<i>Content (text)</i>					
<i>Content (objects)</i>					
<i>Header metadata</i>					
<i>Citation metadata</i>					
<i>Organization (ToC)</i>					
<i>Terms</i>					
<i>Named Entities</i>					
<i>Concepts</i>	x (indirectly)				
<i>Topics</i>				x (LDA)	x (LDA)
<i>Relations</i>	x (prerequisite)	x (prerequisite)	x (prerequisite)		
Output	graph	-	graph	-	-
Knowledge model	no	no	no	no	no

Table 2.12: Detailed information about the analyzed papers (11/11).

	Larrañaga et al. [164]	Fiallos & Ochoa [99]	Thareja et al. [262]
Cluster	C4	C4	C4
Year	2014	2019	2022
Input	textbooks	general documents	textbooks
Methods	rule-based, NLP-based, learning-based, semi-automatic	rule-based, NLP-based, semantic-based	learning-based, knowledge-based
External KB			Wikipedia
<i>Layout</i>		x	x
<i>Content (text)</i>	x (partial)	x	x (partial)
<i>Content (objects)</i>			
<i>Header metadata</i>			
<i>Citation metadata</i>			
<i>Organization (ToC)</i>	x	x	x
<i>Terms</i>	x (index)	x (content)	x (index)
<i>Named Entities</i>			
<i>Concepts</i>			
<i>Topics</i>		x (LDA)	
<i>Relations</i>	x (part-of, is-a, prerequisite, and next)		x (prerequisite)
Output	ALOCOM (XML)	ontology	graph
Knowledge model	no	no	no

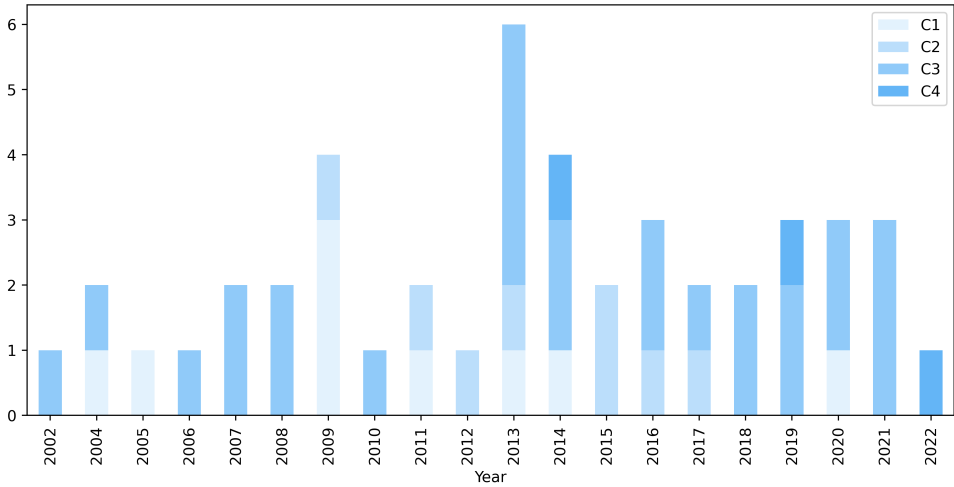


Figure 2.1: Number of approaches by year and cluster.

the 2000s (Chao & Fan [52] and Hassan [127]). During the 2010s, extraction of specific content objects (Oro & Ruffolo [215]), as well as header and citation metadata (Kern et al. [151] and Kern & Klampfl [152]) can be observed. **C2** approaches build on top of **C1** approaches and are centered around 2014.

The diversity of approaches in **C3** spans 20 years of research. Extraction of terms (Larrañaga et al. [165]) and domain concepts (Xu et al. [294]) attracts attention since the early 2000s. Linking of Named Entities were studied as early as 2007 (Mihalcea & Csomai [195]), and continue to be a topic of interest nowadays (Aghae-brahimian & Cieliebak [5]). Extraction of concept relations are more studied since 2016 (Chaplot et al. [53] and Labutov et al. [161]). Finally, techniques for topic extraction belong to the last decade (Guerra et al. [117] and Kawamata et al. [147]).

Larrañaga et al. [164] is the earliest approach in **C4**, which dates back to 2014. However, the novelty of knowledge extraction across all element categories is reflected by the fact that the remaining approaches have been developed in the last three years (Fiallos & Ochoa [99] and Thareja et al. [262]). As seen in Figure 2.1, bars get darker with time, indicating an overall shift towards more knowledge-oriented extraction.

Input

The analyzed approaches were designed to work on six different types of textual resources: general documents, textbooks, scientific articles, books, software-requirements documents, and MOOCs. Figure 2.2 shows the distribution of the approaches for each resource type. The majority of the studied approaches works with general documents; therefore, they can also be applied to textbooks. The second largest group corresponds to textbook-specific approaches. The third position corresponds to approaches for scientific articles. This kind of resource shares content and structure elements with textbooks. A few approaches work with

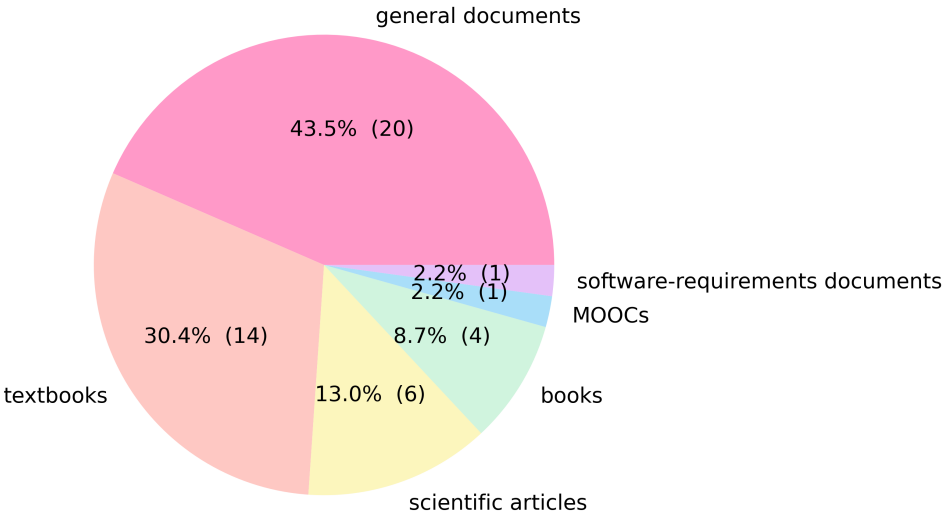


Figure 2.2: Distribution of approaches by type of resources.

general books, which are a more abstract representation of textbooks. Finally, in the last two positions there are one approach specific for software-requirements documents and another for MOOCs. Those two types of resource share domain elements with textbooks.

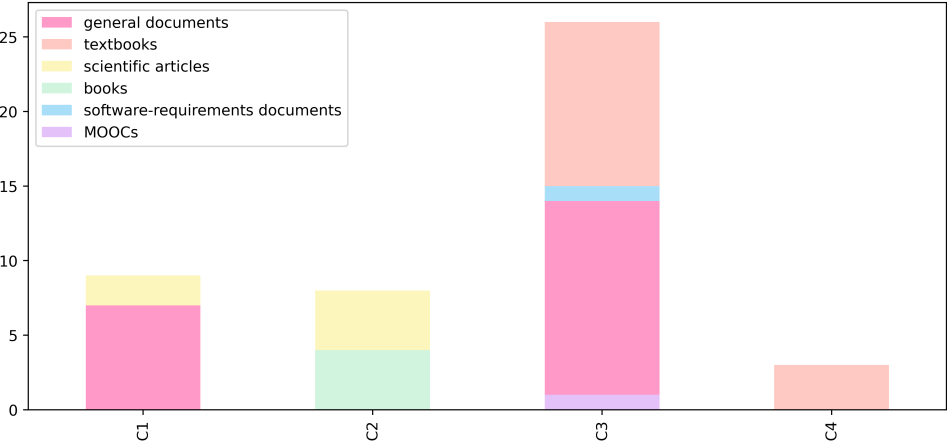


Figure 2.3: Distribution of approaches per cluster and type of resources.

Figure 2.3 analyzes the individual groups. General documents have been the focus of approaches in C1. Since content extraction is a common task for any textual

resource, the trend is logical. **C2** groups approaches that analyze the table of contents, which are a typical component of any book. Additionally, scientific articles have been the subject of methods that extract the hierarchical structures using section headers. Once more, it is possible to observe the diversity of approaches in **C3**—this time, because of the different kinds of analyzed documents. For example, extraction of terms has been applied to general documents (Medelyan et al. [189] and Yuan et al. [296]), textbooks (Chaudhri et al. [55] and Wang et al. [285]), and software-requirement documents (Dwarakanath et al. [82]). Finally, approaches in **C4** focus on textbooks exclusively.

Methods

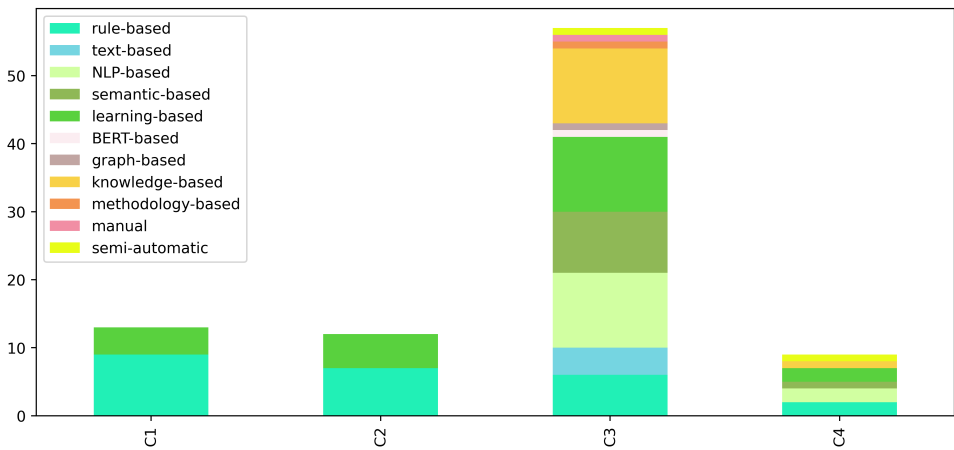


Figure 2.4: Number and types of methods by cluster.

In general, various methods are used to extract the different elements. Figure 2.4 shows the number of times that the different kinds of methods were used in each cluster. Approaches in **C1** and **C2** use rule-based and learning-based approaches exclusively. Since different approaches in **C3** extract different elements, in this cluster, all 11 kinds of methods are used in different ways. Finally, **C4** also displays a diversity of methods since the approaches in this cluster focus on specific elements across all three categories. In terms of combinations of methods, 17 approaches use only one main method, while 29 rely on a combination of two or more.

Rule-based systems are used to extract layout and content knowledge. Across multiple approaches, rules and heuristics are used to extract the textual content from the low-level PDF objects. Figure 2.5 shows the rules and process for text segmentation used in Chao & Fan [52]. The pipeline shows the general use of rules to form words, lines, and paragraphs based on the position of characters, inline spacing, and between-line spacing. The same kind of rules is applied in more recent approaches, like the one by Tkaczyk et al. [265]. Despite the popularity and accuracy of machine learning approaches, rule-based approaches stay highly accurate. In 2017, Bast & Korzen [26] presented a rule-based approach based on

distances, positions, and fonts of characters. The approach outperformed 13 PDF extraction tools.

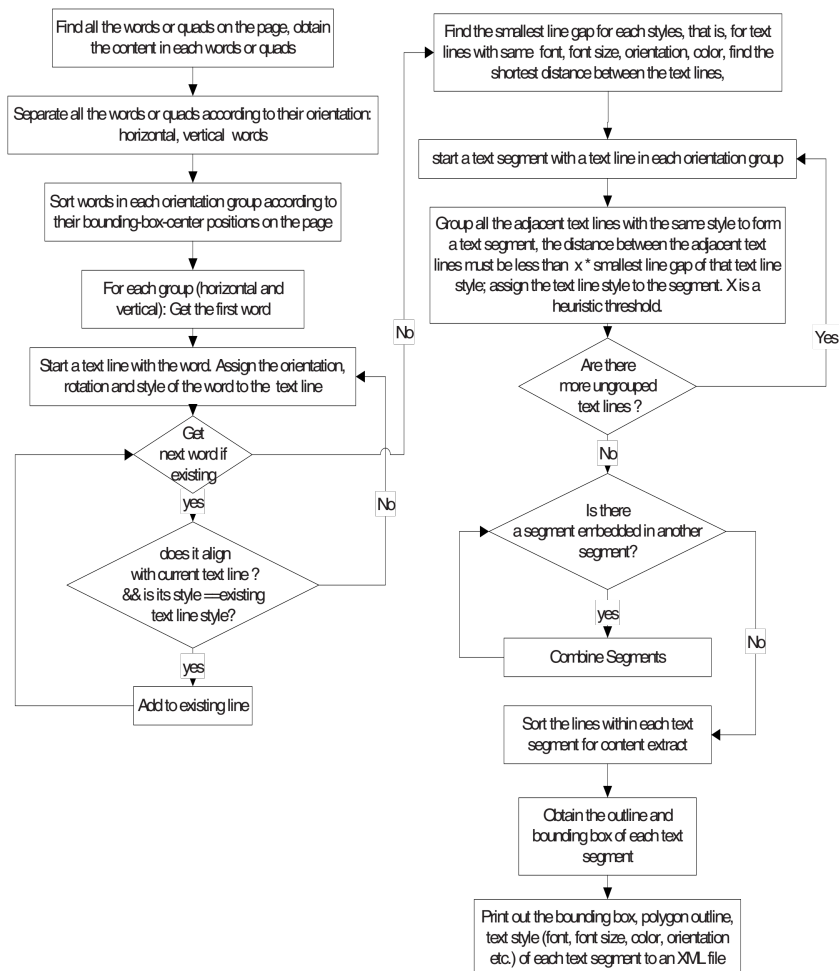


Figure 2.5: Text segmentation process used in Chao & Fan (image from [52]).

A combination of rule-based and learning-based approaches has been used to recognize specific text blocks like formulae (Lin et al. [168]) and metadata (Kern & Klampfl [152]). For example, Tkaczyk et al. [265] used two supervised classifiers: a Support Vector Machine (SVM) to classify the document's zones into four main categories and a Conditional Random Field (CRF) to identify the label of tokens for reference strings, as shown in Figure 2.6. The example shows that seven different metadata tokens are identified in the bibliographic reference. In the same workflow, rules were used to extract the correct reading order and header metadata from labeled zones.

Organizational elements such as section headers and the ToC have been extracted

[9] L. O'Gorman. The document spectrum for page layout analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1162–1173, 1993.

Figure 2.6: Example of reference tokens recognized using a supervised classifier in Tkaczyk et al. (image from [265]).

using either a stand-alone rule-based method or in combination with machine learning. Ramanathan et al. [226] presented an approach based only on rules and regular expressions to identify ToC entries. Wu et al. [293] applied rules for entry detection according to three different ToC styles, as shown in Figure 2.7. In a *flat* ToC, all entries have the exact left alignment, while in a *ordered* ToC, the entries start with a section number, and in a *divided* ToC, entries are divided into blocks. The hybrid approach from Tuarob et al. [270] is based on machine learning to recognize section boundaries and rules to identify the hierarchy of sections. The 22 features used to characterize section headers in this last approach are shown in Figure 2.8. In the approach, the features are divided into three groups that capture the formatting and structural properties of scientific documents: pattern-based (PAT), style-based (STY), and structured-based (STR).

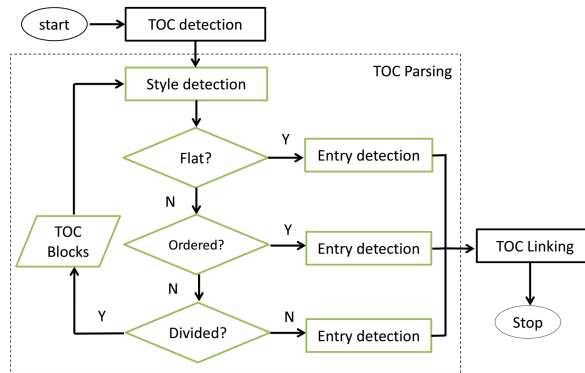


Figure 2.7: ToC recognition process proposed by Wu et al. based on three different styles (image from [293]).

Extraction of terms from the back-of-the-book index has been done primarily using rule-based methods to handle the different characteristics of such indices (Wali et al. [280]). Extraction of terms using the content of local or external resources has been done with a diversity of methods. Natural Language Processing techniques are handy for identifying terms and their relations. For example, Larrañaga et al. [165] used part-of-speech information in combination with heuristics to identify structural and sequential relationships between index terms. Figure 2.9 shows an example of the pattern *noun + adjective*, *noun + noun phrase*, which identifies *is-a* relations. In the figure, the *agent* noun appears in both the top-level and secondary-level terms, indicating that *mobile agents* is-a *agent*. The approach by Dwarakanath et al. [82]

Grp	Feature	Description
PAT	IS SEC HEADER W/ NUM	Whether the line matches the number-leading section header pattern.
	IS UPPER SEC HEADER W/ NUM	Whether the above line matches the #-leading section header pattern.
	IS LOWER SEC HEADER W/ NUM	Whether the lower line matches the #-leading section header pattern.
	IS SEC HEADER W/O NUM	Whether the line matches the section header without number pattern.
	IS UPPER SEC HEADER W/O NUM	Whether the above line matches the section header without pattern.
	IS LOWER SEC HEADER W/O NUM	Whether the lower line matches the sec. header without number.
	IS STANDARD SEC	Whether the line is one of the standard section headers.
	IS CAPTION	Whether the line is a caption or a figure, table, or algorithm.
STY	MODE FONTSIZE	Mode fontsize (in Pt.) of all the characters in the line
	FRAC FONT MODE TO DOC AVG	Fraction of the mode fontsize in the line to the avg. fontsize in the doc.
	FRAC FONT MODE TO DOC MODE	Fraction of mode fontsize in the line to the mode fontsize in the doc
	FRAC UPPER GAP TO MODE GAP	Fraction of the gap space between the line and the upper line (in cm) to the mode gap space between two lines in the document.
	FRAC LOWER GAP TO MODE GAP	Fraction of the gap space between the line and the lower line (in cm) to the mode gap space between two lines in the document.
	FRAC UPPER GAP TO AVG GAP	Fraction of the gap space between the line and the upper line (in cm) to the average gap space between two lines in the document.
	FRAC LOWER GAP TO AVG GAP	Fraction of the gap space between the line and the lower line (in cm) to the average gap space between two lines in the document.
	ARE ALL CHARS BOLD	Whether all the characters in the line have boldface font style.
	ARE ALL WORDS CAPITALIZED	Whether all the words in the line are capitalized.
	FRACTION NUMWORDS TO AVG NUMWORDS	Fraction of the number of words in the line to the average number of words per line of the document.
STR	IS AFTER ABS INT	Whether the abs or int sections has already been detected.
	IS BEFORE REF	Whether the reference section header has NOT yet been detected.
	IS FIRST LINE OF PAGE	Whether the line is the first line of page.
	IS LAST LINE OF PAGE	Whether the line is the last line of page.

Figure 2.8: Features used by Tuarob et al. to characterize section headers (image from [270]).

3.4.2. Agents
3.4.2.1. Mobile Agents
3.4.2.2. Stationary Agents
...

Figure 2.9: Example of a part-of-speech pattern used to recognize *is-a* relations in Larrañaga et al. (image fragment from [165]).

extracts simple and compound terms using n-grams, morphosyntactic patterns, and noun phrases. Learning-based methods have also been used to extract the same elements. Chaudhri et al. [55] used the terms from the back-of-the-book indices to fine-tune a BERT model for term extraction from unseen textbooks. The approach by Yuan et al. [296] is based on the Seq2Seq model with attention to generate a variable number of keyphrases. Figure 2.10 show the architecture of the proposed model, in which, multiple phrases are concatenated into single sequences using the *sep* delimiter.

Knowledge-based methods use knowledge bases to extract possible terms and link them to named entities. Titles of Wikipedia articles have been used in multiple approaches as candidate terms (Medelyan et al. [189] and Wang et al. [285]). Mihalcea & Csomai [195] used Wikipedia as a resource for automatic keyword extraction and word sense disambiguation. Their approach limits keywords to those that have a valid corresponding Wikipedia article. Wikipedia links are used for both keyword ranking and linking to the right Wikipedia article. Another popular KB for extracting domain elements is DBpedia. Mendes et al. [192]/Daiber et al. [63] exploited the graph of labels, links, synonyms, and alternative and ambiguous names to se-

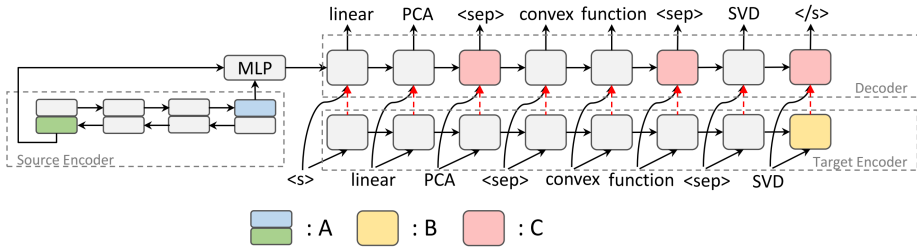


Figure 2.10: Architecture proposed by Yuan et al. for keyphrase generation. In the proposed training paradigm, multiple phrases are concatenated into a single sequence with a delimiter `<sep>`. Once the model have finished generating phrases, it outputs a special token `</s>` to terminate the decoding process. A: last states of a bi-directional source encoder; B: last state of target encoder; C: decoder states for delimiters or end-of-sentence tokens (image from [296]).

lect the phrases that may indicate a mention of a DBpedia entity. A Vector Space Model (VSM) resolves ambiguities and links the phrases to the correct named entities. Zhu & Iglesias [302] used categories and textual descriptions of entities in DBpedia as semantic features to apply semantic-based, NLP-based, and learning-based methods for Named Entity Disambiguation. Part of their framework is presented in Figure 2.11. Once terms have been linked to a named entity in a KB, it is possible to extract the definitions contained in such a KB.

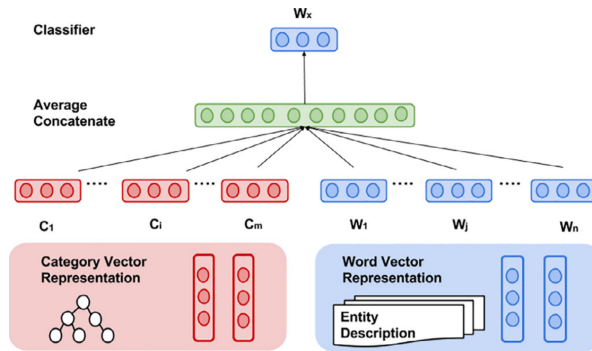


Figure 2.11: Architecture for jointly learning embeddings of word and category through entities in a KB for linking Named Entities, as proposed in Zhu & Iglesias (image from [302]).

Wang et al. [283] acknowledged that automatic methods for concept annotation have insufficient accuracy and proposed a systematic textbook manual annotation methodology based on six steps to guide the coding procedure of domain concepts, which is presented in Figure 2.12. Since automatic identification of the domain concepts is a challenging task, multiple methods are usually applied together. Chau et al. [54] presented a machine learning approach based on linguistic (NLP-based), statistical (text-based), and external resources (knowledge-based) features to identify the most relevant concepts concerning a target domain or a related domain. A

combination of ruled-based and semantic-based techniques was used by Kida et al. [153] to assess the degree of domain specificity of terms. Figure 2.13 presents an example of their approach, where three terms are evaluated according to the domain of a corpus collected from the Web.

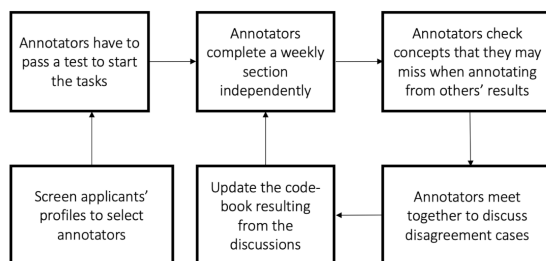


Figure 2.12: Systematic textbook annotation methodology developed by Wang et al. (image from [283]).

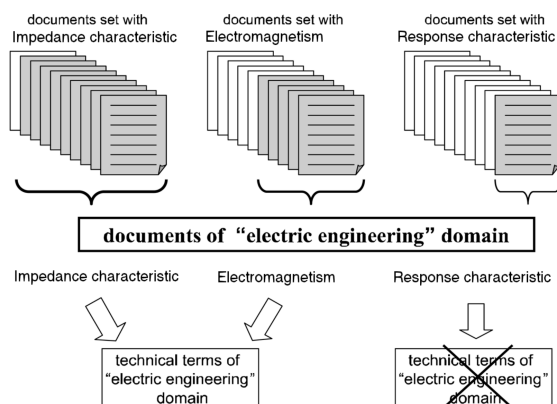


Figure 2.13: Example from Kida et al. of the degree of specificity of a term based on the domain of the documents (image from [153]).

Concept relations have been extracted using learning-based and rule-based methods. Labutov et al. [161] used textbooks as a source of author-encoded knowledge to create an outcome/prerequisite classifier. Adorni et al. [4] presented an approach based on the co-occurrence of concepts and the structure of the textbooks. Latent Dirichlet Allocation (LDA), a generative statistical model used in NLP has been the primary method of topic modeling in documents (Guerra et al. [117] and Kawamata et al. [147]).

Finally, different combinations of all the discussed methods have been used by the approaches that extract knowledge across the three categories of elements. Larrañaga et al. [164] presented a hybrid approach using NLP techniques, heuristic reasoning, and ontologies for the knowledge extraction process. Figure 2.14 shows the proposed process. Fiallos & Ochoa [99] proposed to parse documents to ex-

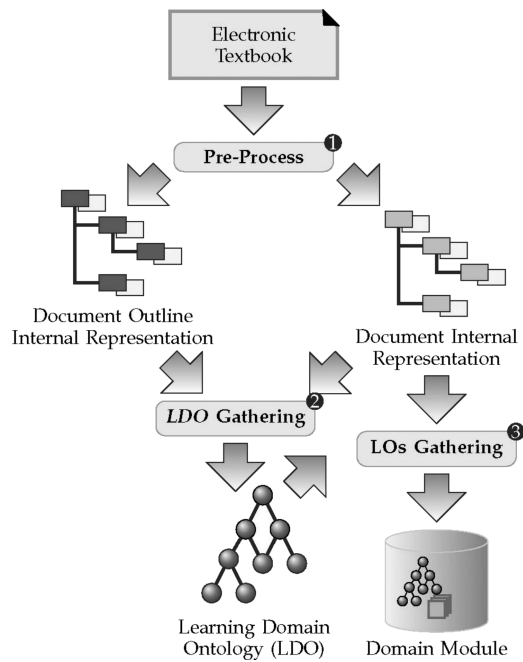


Figure 2.14: General approach from Larrañaga et al. (image from [164]).

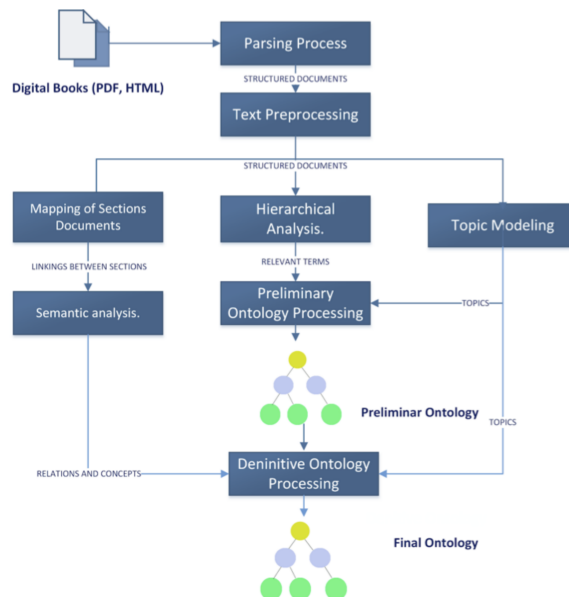


Figure 2.15: General approach from Fiallos & Ochoa (image from [99]).

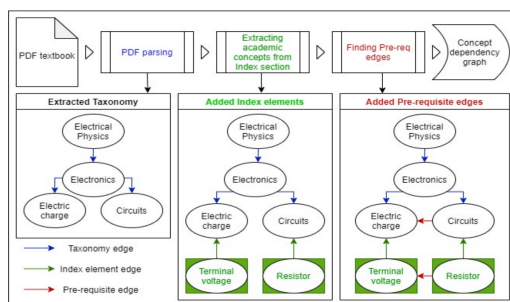


Figure 2.16: General approach Thareja et al. (image from [262]).

tract concepts and relations using the titles of the sections and topics inferred using LDA. Their approach is presented in Figure 2.15. Thareja et al. [262] used machine learning with geometrical and Wikipedia features to parse the textbooks and extract concepts and prerequisite relations, as shown in Figure 2.16.

External KB

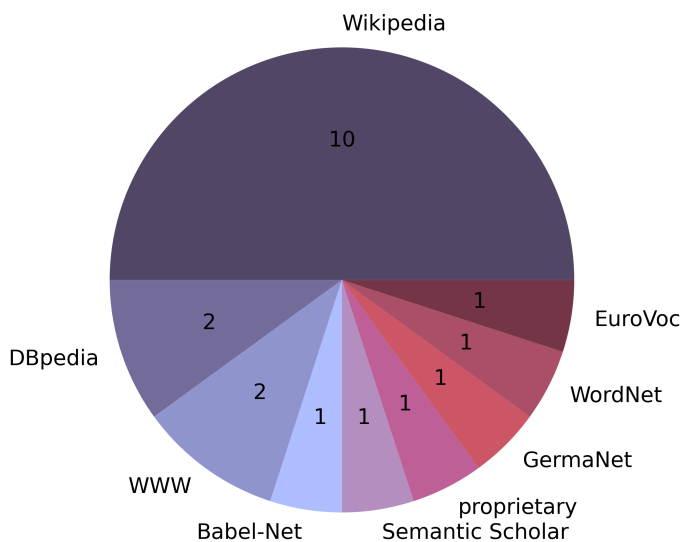


Figure 2.17: Number of approaches according to the external KB that they use.

External KBs (one or multiple) are used by approaches in **C3** and **C4**. Extraction of content and structure elements do not require external knowledge since the tasks can be done using only local information. Figure 2.17 shows the employed KB and the number of approaches that used them. Wikipedia is used in 10 different approaches, making it the most popular KB in the analyzed set. DBpedia, a Knowledge

Graph constructed from Wikipedia, is used in two approaches. The World Wide Web (WWW) follows with two occurrences. Even though, technically, the WWW is not a KB, it is used as a repository of content to identify domain-specific concepts (Kida et al. [153] and Rigutini et al. [229]). Other KBs are used once each: BabelNet, Semantic Scholar, GermanNet, and WordNet. Finally, EuroVoc, a multilingual thesaurus of the European Union, is used in one of the approaches.

Output

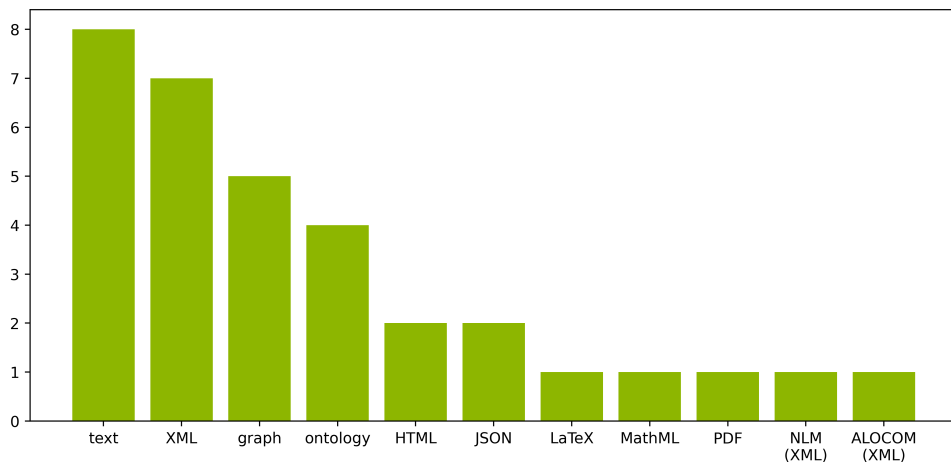


Figure 2.18: Number of approaches by output format.

From the analyzed approaches, 27 provide information about the output formats used to represent the extracted knowledge from the textual resources. The choice of a format is of the utmost importance since it should facilitate both human and machine readability. Figure 2.18 shows the used formats and the number of approaches that utilize them. The most important knowledge representation mechanism is XML, either with a custom definition (in seven approaches) or following a specific standard (two approaches). Figure 2.19 shows how in Mendes et al. [192] the identified named entities are represented using XML.

Text representation is widely used with eight occurrences. Figure 2.20 shows how caption text and reference sentences associated with algorithmic procedures are represented in Tuarob et al. [269]. Graph and ontology representations are the third and fourth most used output formats, respectively. Figure 2.21 shows the received input and the proposed output in Wang et al. [284]. HTML, JSON, and LaTeX are other used formats.

Knowledge Model

Approaches in C4 extract elements across all three defined categories. Nonetheless, the produced models only contain partial information.

Larrañaga et al. [164] extract knowledge about the content, structure, terms, topics, pedagogical relationships, and learning objects. However, the produced model

```
<Annotation text="Pop star Michael Jackson..."
confidence="0.3" support="30"
types="Person,Place,...">
<Resources>
  <Resource URI="dbpedia:Michael_Jackson"
support="5761"
types="MusicalArtist,Artist,Person"
surfaceForm="Michael Jackson" offset="9"
similarityScore="0.31504717469215393" />
  ...
</Resources>
</Annotation>
```

Figure 2.19: Example of how extracted named entities are represented using XML in Mendes et al. (image from [192]).

Pseudo-code
<pre> procedure INIT-PDA {<i>Invoked when the router comes up.</i>} begin Initialize all tables; call PDA; end INIT-PDA procedure PDA {<i>Executed at each router i. Invoked when an event occurs</i>} begin (1) call NTU; (2) call MTU; /* Updates T^i */ (3) if (there are changes to T^i) then Compose an LSU message consisting of topology differences using <i>add</i>, <i>delete</i> and <i>change</i> link entries; endif (4) Within a finite amount time, send the LSU message to all neighbors; end PDA </pre>
Figure 1: The Partial-topology Dissemination Algorithm
Caption Text
Figure 1: The Partial-topology Dissemination
Reference Sentences
<p>The INIT-PDA procedure in Fig. 1 initializes the tables of a router at startup time; all variables of type distance are initialized to infinity and those of type node are initialized to null. Similarly, in Fig. 10, the delays obtained using MP routing for NET1 are within 28% envelopes of delays obtained using OPT routing. Fig. 11 compares the average delays of MP and SP for CAIRN. In Fig. 12, for NET1, MP routing performs even better; average delays of SP are as much as five to six times those of MP routing which is due to higher connectivity available in NET1. For CAIRN, Fig. 13 show the effect of increasing T_1 when T_s and the input traffic is fixed. Similarly, for NET1, delays for SP increased significantly while there is negligible change in delays of MP as can be observed in Fig. 14, respectively. This becomes evident in Fig. 15, which shows a typical response in NET1 when the flow rate is a step function (i.e., the flow rate is increased from 0 to a finite amount at time 0). Fig. 18 shows the delays for SP and MP.</p>

Figure 2.20: Example of textual representation in Tuarob et al. (image from [269]).

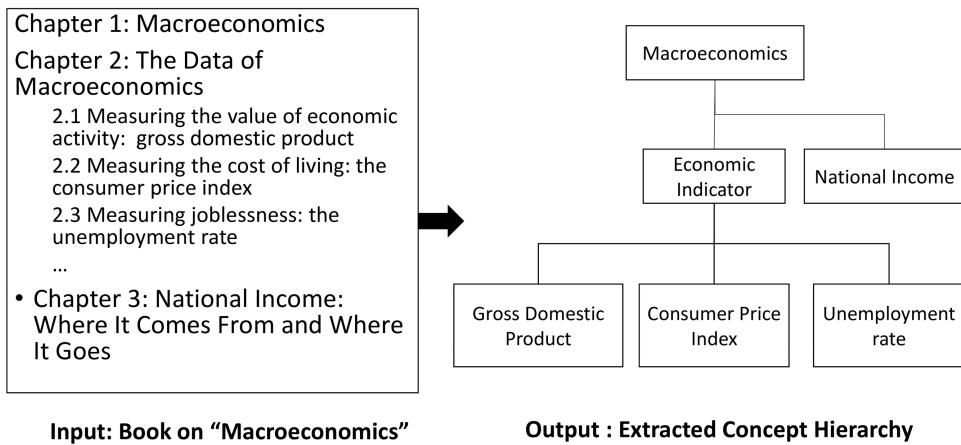


Figure 2.21: Graph output representation used in Wang et al. to represent the extracted knowledge (image from Wang et al. [284]) .

only contains information about the topics, pedagogical relationships, and learning objects in the textbooks. Similarly, Fiallos & Ochoa [99] output a domain ontology that focuses on topics and their semantic relationships. The workflow by Thareja et al. [262] also starts with the parsing of the textbook to extract the content, then the organization hierarchy, moving to index elements, and finalizing with prerequisite information. However, the content knowledge is not represented in the final concept dependency graph.

2.4 Other Dimensions

This section looks at the approaches using three additional dimensions that were observed after indexing the papers.

2.4.1 Unstructured and Semi-structured Resources

This analysis included unstructured and semi-structured information resources because the object of interest in this thesis is PDF textbooks. After the analysis, it was noticed that each kind of resource is related to a specific stage of the knowledge extraction process. The approaches that extract content and structure information work with unstructured resources. PDF or TXT documents rarely preserve information about their structure². Therefore, formatting and organization cues are used to extract the knowledge from the resources. Once the target knowledge has been extracted, it is usually represented using a semi-structured format, such as XML (Oro & Ruffolo [215]). The input of approaches that extract domain knowledge is usually a semi-structured resource. Such resources contain the necessary content elements in a structured and readable format. For example, Lopes et al. [171]

²PDF tags provide logical structures, but they are rarely used.

rely on an external tool to obtain an XML file with the document's words and their morphological characteristics before the extraction of keyphrases begins.

2.4.2 Textbook's Encoded Knowledge

Different approaches have recognized the value of the implicit knowledge encoded in textbooks. The approach from Labutov et al. is based on the assumption that "textbooks naturally encode experts' knowledge implicitly via the structure of the books they write" [161]. Wang et al. claimed that "textbooks provide organized units of knowledge and a balanced and chronological presentation of information" [284]. Other approaches (Larrañaga et al. [165] and Wali et al. [280]) leverage the terms from the back-of-the-book index as a source of domain knowledge. The motivation for Guerra et al. [117] is the abundance of online textbooks and how links among them can be created to increase access to different representations of the same topic.

2.4.3 Language of Documents

The majority of the approaches work with documents written in English. However, other languages are represented as well. Fang et al. [93] and Gao et al. [110] work with Chinese books. Larrañaga et al.'s work has been on documents written in Basque language [164, 165]. Linking of Named Entities in Mendes et al. [192] was originally developed only for the English language, but the approach was later extended to work with nine additional languages (Daiber et al. [63]).

2.5 Limitations

This chapter has analyzed relevant approaches for knowledge extraction from textual resources, but it is not a systematic literature review. Therefore, other approaches dealing with the same scope may have been missed. However, it was not possible to find more approaches in C4 (knowledge extraction across all three categories) than those described. Additionally, the coding was done only by one person (the author of this thesis). Double coding would have been preferred to discuss and adjust the annotations if necessary.

Another limitation is that the selection of textbook elements covers multiple research topics, and they can be analyzed from different perspectives. For example, Martinez-Rodriguez et al. [184] presented a literature survey over similar elements from the perspective of Information Extraction techniques in a Semantic Web setting.

2.6 Conclusions and Future Work

In this chapter, 46 approaches for knowledge extraction from unstructured and semi-structured textual resources have been analyzed. Six features have been used to present a general overview of the approaches. One combined feature described the approaches in terms of 11 textbook elements that can be extracted from tex-

tual resources. Also, the main findings have been reported. There is a diversity of approaches that extract knowledge from different textual resources, use multiple methods, integrate information with external knowledge bases, and represent their outcomes using different formats. Three approaches extract elements across all defined categories, however, none of them produces a complete TKM.

Extracting valuable knowledge from different textual resources, not only textbooks, has stayed active for at least 20 years, and new approaches keep emerging. Integrating extracted knowledge from multiple elements and representing that knowledge in a format that provides valuable insights into specific domains is still an open research area. The lack of an approach that presents a unified approach towards automated extraction of full-scale textbook models motivates the research presented in this thesis. The approach described in Chapters 3-5: (1) extracts knowledge across content, structure, and domain aspects of textbooks; 2) encodes all the extracted knowledge using a human and machine-readable format; and (3) creates knowledge models with multiple applications in systems that require formal knowledge representations. The proposed approach leverages the idea that the high-quality knowledge encoded in textbooks by their authors can be extracted automatically.

As future work, the presented analysis will be extended by formally performing a systematic literature review on the same topic.

CHAPTER 3

Order out of Chaos: Construction of Knowledge Models from PDF Textbooks

Abstract Textbooks are educational documents created, structured and formatted by domain experts with the main purpose to explain the knowledge in the domain to a novice. Authors use their understanding of the domain when structuring and formatting the content of a textbook to facilitate this explanation. As a result, the formatting and structural elements of textbooks carry the elements of domain knowledge implicitly encoded by their authors. This chapter presents an extendable approach towards automated extraction of this knowledge from textbooks taking into account their formatting rules and internal structure. The focus is on PDF as the most common textbook representation format; however, the overall method is applicable to other formats as well. The evaluation experiments examine the accuracy of the approach, as well as the pragmatic quality of the obtained knowledge models using one of their possible applications—semantic linking of textbooks in the same domain. The results indicate high accuracy of model construction on symbolic, syntactic and structural levels across textbooks and domains, and demonstrate the added value of the extracted models on the semantic level.

This chapter is based on the following publications:

Alpizar-Chacon, I. & Sosnovsky, S., “Order out of chaos: construction of knowledge models from pdf textbooks”, in: *Proceedings of the ACM Symposium on Document Engineering 2020*, 2020, pp. 1–10.

Alpizar-Chacon, I. & Sosnovsky, S., “Knowledge models from pdf textbooks”, *New Review of Hypermedia and Multimedia*, vol. 27, no. 1-2, 2021, pp. 128–176.

3.1 Introduction

Textbooks are high-quality textual resources. Content of a typical textbook is characterized by:

- focus: belongs to a narrow, cohesive domain;
- quality: is created and curated by domain experts;
- purpose: consists primarily of expository text explaining domain knowledge to a novice.

For the purpose of text analysis and information extraction, textbooks have often been considered as high-quality but non-structured information resources [227]. In fact, good-quality textbooks are not just collections of texts in the same domain. Methodologies of writing a good textbook seek to facilitate learning not only by making the text itself easier to understand, but also by employing logical structure and formatting signals across its content [49]. It has been shown that text structure awareness is an important foundation for improving text comprehension and recall by learners [122].

Hence, the content of a good textbook is structured into chapters and subchapters that are provided with informative headings. It is formatted in a consistent manner that attracts attention to important fragments and reduces information clutter. It includes browsing (ToC) and searching (index) aid, and is ordered from basic concepts to more complex notions. Authors use their understanding of the domain when placing these formatting and structural cues in a textbook; and they do this with a primary purpose—to facilitate explanation of domain knowledge to a novice. As a result, the formatting and structural elements of a textbook reflect its domain semantics. The question is—can such knowledge be automatically extracted from textbooks? And if it is extracted and formally represented as machine-readable knowledge models, what would be the quality and the value of such models?

This chapter seeks to answer both questions by presenting and evaluating a unified approach towards automated extraction of textbook models from their formatting principles and internal structure. The proposed method focuses mainly on PDF as the most common (and more challenging) format for representing digital textbooks; however, the overall approach is also applicable to other formats that are more explicit and coherent in their structural specifications than PDF (e.g., EPUB textbooks have been processed as well). A typical PDF document contains thousands of low-level objects, multiple compression mechanisms, different font formats, lines, curves, vectors, and ancillary content [287]. The objects inside a content stream in a PDF are instructions to draw shapes, images, and text, which can appear in any sequence. Unfortunately, PDF textbooks rarely preserve information that would facilitate extraction of their structure (i.e., letters composing words, words belonging to paragraphs, organization of lists, tables and figures, hierarchy of sections, or even the reading order of the text) [265]. The PDF standard does allow expressing a logical structure of a document with tags, but their usage is seldom and inconsistent; therefore, the proposed approach does not rely on tagging. Instead, in this chapter, an approach that captures common practices of textbook formatting and

organization has been developed. While two different textbooks are likely to differ in their formatting styles; within a single textbook, formatting follows a strict set of principles. And when formatting and structuring conventions across textbooks are compared, it is possible to see very common patterns. This chapter's approach formalizes these patterns as rules and applies them according to a unified process that consists of four main stages: extraction of text, role labeling of fragments, identification of organizational elements, construction of a model.

The main contributions of this chapter are: (1) a general approach for extracting knowledge models of textbooks; (2) implementation of the approach specifically for the PDF format; and (3) evaluation of the accuracy and added value of the proposed approach.

This chapter is organized as follows. Section 3.2 presents related work. The proposed approach is described in Section 3.3. Section 3.4 presents and discusses the evaluation. Finally, Section 3.5 concludes.

3.2 Related Work

Technologies for automated analysis of PDF documents have been developed before primarily to facilitate retrieval of PDF documents and their conversion to other formats. In principle, such an analysis can be performed on different levels of information extraction from symbolic (text and layout) to structural (metadata and organization), to semantic (knowledge models).

3.2.1 Text and Layout

Chao & Fan [52] proposed to separate documents into text, image, and vector graphics layers. On the text layer, words are obtained from symbols and then merged into lines and segments, while objects recognized on the image and vector layers are saved separately. Hassan [127] used a similar bottom-up approach to recognize an object hierarchy by directly reading the content stream of the PDF and identifying and merging small textual objects into bigger ones: from characters to words, to lines, to fragments. There are several tools available to extract text from PDF documents, e.g.: pdftotext¹, PDFBox², GROBID³, and PdfAct⁴. Bast & Korzen [26] evaluated 14 state-of-the-art text extraction tools with respect to four aspects: identification of paragraph boundaries, distinction between body text and non-body text, understanding of reading order, and detection of word boundaries.

3.2.2 Content Objects

A lot of literature has focused on identification and extraction of specific types of content objects in scholarly articles and books. Kern et al. have developed several methods for detecting bibliography metadata by classifying text blocks and words

¹<https://www.xpdfreader.com/pdftotext-man.html>

²<https://pdfbox.apache.org/>

³<https://github.com/kermitt2/grobid>

⁴<https://github.com/ad-freiburg/pdfact>

within blocks [151, 152]. The CERMINE system [265] uses both textual and geometric features to classify references. CiteSeerX [290] relies on word-specific and line-specific features to recognize fifteen unique fields for metadata and citation extraction [61]. Tables [93, 215, 290], diagrams and figures [108, 247], mathematical formulae [23, 169], algorithms [269], and pseudo-code fragments [268, 270] have been extracted using rule-based and machine learning techniques.

3.2.3 Document Organization

ToCs were often used as a source to recognize hierarchical organization of documents. Gao et al. [110] applied clustering to learn the layout model for ToC entries based on such features as the number of word blocks, font size, and height of each line. Ramanathan et al. [226] and Wu et al. [293] extracted ToCs and bookmarks from documents with the help of rules and regular expressions. Tuarob et al. [270] also used regular expressions to recognize standard sections, a machine learning approach to identify arbitrary sections, and a rule-based strategy to build a hierarchical structure of documents without a ToC. Unlike ToCs, textbook indices surprisingly have not yet received much attention in the literature. One found example uses the index section to extract terminology of a single book by applying NLP tools and heuristic reasoning [165].

3.2.4 Knowledge Models

Research on creating complete knowledge models from textbooks has been limited. Larrañaga et al. [164] described a system that uses NLP techniques, heuristic reasoning, and ontologies for the semiautomatic construction of a representation of the knowledge to be learned from electronic books. Wang et al. [284] have extracted concept hierarchies from textbooks based on their ToCs. Also, Sosnovsky et al. [254] experimented with harvesting topic-based models from HTML textbooks using structures of their headings and mapping them into ontologies to facilitate more fine-grained personalization of the textbook content. The work presented in this chapter falls into this category—an approach implementing all stages of textbook knowledge extraction.

3.3 Approach

The approach extracts a knowledge model from a textbook using its symbolic, structural and semantic elements. Then, the knowledge model is serialized as an XML file using the Text Encoding Initiative (☞ Section 3.3.5). Figure 3.1 shows the overall workflow of the approach, including its four main stages, steps at each stage, and the number of rules associated with each step. Altogether, 38 unique rules have been defined. The workflow is mostly linear gradually moving from rules processing more basic steps to the rules extracting textbook structure and domain knowledge. There are two feedback loops indicating additional information that higher-order rules supply to the lower-order rules to correct and/or improve detection of basic textbook elements.

On the abstract level, this chapter reuses some ideas proposed in the literature for the creation of the rules. For example, the text is reconstructed from the PDF in a bottom-up manner by merging page objects into more significant components similar to Hassan’s approach [127]. Roles of text fragments are inferred from formatting styles present in a textbook, which is related to the work by Gao et al. [108]. To identify hierarchical structures of textbooks, a method similar to the one by Wu et al. [293] for ToC recognition and extraction is used. Also, this chapter’s approach heavily utilizes textbooks’ index sections to extract fine-grained domain terminology [165]. Many rules correspond directly to the application of guidelines defined by the Chicago Manual of Style [263]. Finally, a set of rules is derived directly from examination of a large set of textbooks (e.g., position of page numbers, and missing page numbers). The modular nature of the rule-based approach supports its gradual refinement. Each time a new variation of a formatting or structural pattern is encountered, the approach is extended by modifying an existing rule or adding a new one. To the best of the author’s knowledge, the research presented in this chapter is the first attempt to propose a universal method for the automated extraction of knowledge models from regular textbooks.

3.3.1 General Rules

The workflow has three general rules. `MULTICOLUMN_LAYOUT`⁵ uses the approach of Gao et al. [109] to detect columns in a text by dividing the text area into smaller zones, projecting every character of the text into the zones, and merging neighboring zones. Empty areas with large left and right neighbor zones are marked as column margins. `DEHYPHENATION` follows the approach presented by Kern & Klampf [152] for the dehyphenation of words, which uses lists of hyphenation patterns taken from the TEX distribution⁶ to check if the words separated with a hyphen (-) should be merged into one. Finally, `MATCHING_LINES` is used to search the heading of a section in the lines of a page.

3.3.2 Text Extraction

This subsection introduces several formal definitions to represent the knowledge extracted from textbooks. The notation is also used and expanded in subsequent subsections. T is an input PDF textbook, where $T = (F, C)$. F is the set of all formatting styles in T . Each style $f \in F$ has several attributes (font name, color, size, and extra features - bold/italic). C is a list with all characters in T . Each character $c \in C$ has several geometrical attributes (X- and Y-coordinate on page, rotation angle, width, and height), a Unicode value, and a style f . The purpose of this stage is to parse T and retrieve its content organized according to basic textual elements (words, lines, and pages) using a bottom-up approach. w is a word that corresponds to a list $(c_1, \dots, c_n) \mid \forall c \in C$ formed by characters. The list $W_x = (w_{1x}, w_{2x}, \dots, w_{nx})$ corresponds to a line l_x formed by words. The list $L_y = (l_{1y}, l_{2y}, \dots, l_{ny})$ corresponds to a page p_y formed by lines. Finally, the list $P = (p_1, p_2, \dots, p_z)$ corresponds to all the pages that belong to T and it represents the textual content of the textbook.

⁵Rule names are written in small caps and with underscores separating words.

⁶<https://tug.org/tex-hyphen/>

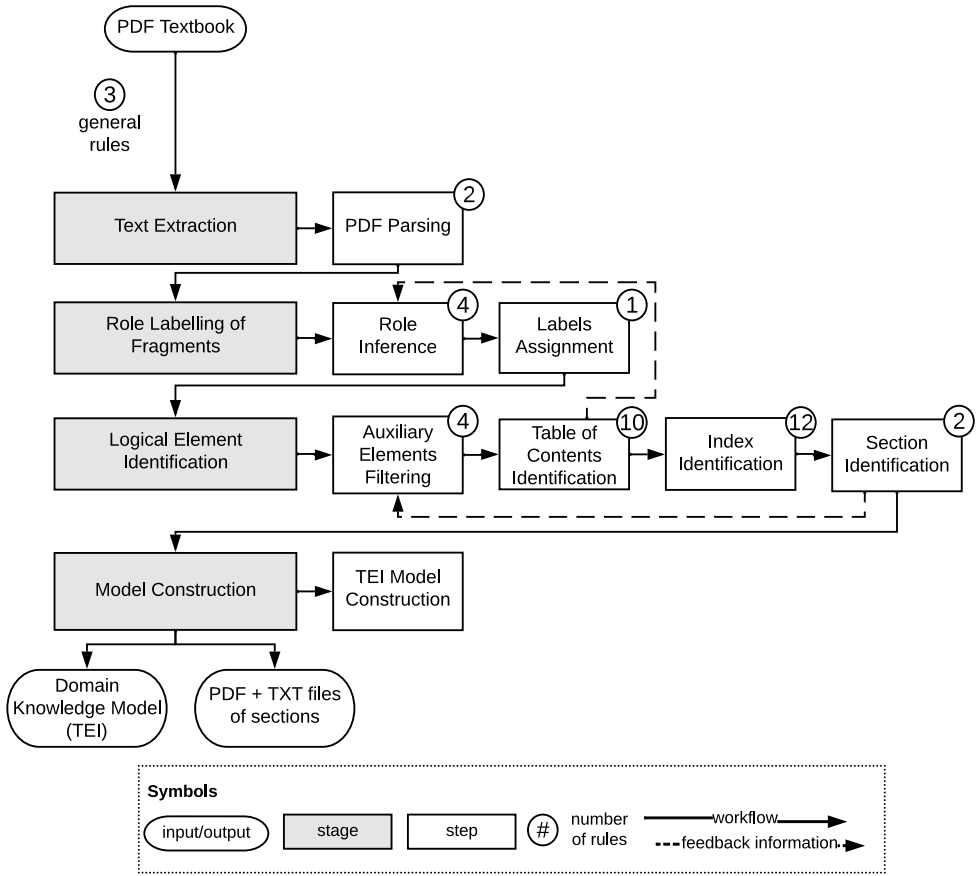


Figure 3.1: Outline of the implemented approach (extraction and formalization phases).

PDF Parsing

During this step, the Apache PDFBox library is used to extract the text from every page of the textbook T . The library processes T as a content stream—character by character—and merges smaller textual objects into bigger ones: characters (c) are grouped into words (w), words into lines (l), and lines into pages (p). Even though the library sorts C according to their positions on a page (left to right and top to bottom), multiple problems were detected in the development of the proposed approach when extracting subscripts, superscripts, formulae, and rotated text. Therefore, two additional rules to process these cases are used. First, `ROTATED_COORDINATES` detects continuous characters in the stream that have the same non-zero rotation angle and applies a linear transformation to rotate them to a horizontal position. Then, `ORDER_OF_CHARACTERS` uses custom bounding boxes (where the bottom of the box corresponds to the baseline of the glyph) for each c to sort C properly. Specifically, $C = (C, \leq)$ such that \leq is a relation that sorts C . The sorting relation first groups the characters according to the pages where they appear.

Then, for each page, the characters are grouped into lines using the bottom coordinates of the bounding boxes. When characters from different lines vertically overlap, the algorithm checks if the lines need to be merged. Finally, lines are merged if the overlapping characters from different lines are next to each other according to their X-coordinates and if their font sizes are different (which is the case for sub- and superscripts). Once C is properly sorted, words, lines, and pages are formed according to PDFBox's own rules to create the list P . At this stage, textual content in P may be still not properly ordered due to multi-column layout or floating text boxes, text fragments (blocks) are not recognized, and hyphenated words are not yet merged, because PDFBox does not support such features. The approach handles such cases in the following stages. At the end of this first stage, a simplified textbook model M of T is created: $M = (P)$.

3.3.3 Role Labeling of Fragments

At the second stage, the workflow assigns role labels (section heading, subheading, important text, body text, etc.) to each text fragment. This process facilitates the subsequent recognition of different logical elements of the textbook.

Role Inference

The set F of formatting styles is used to infer which style belongs to which role by comparing them and setting their logical order. First, ORDER_OF_STYLES is used to sort all recognized formatting styles according to their properties: $F = (F, \leq)$ such that \leq is a relation that sorts F . The relation compares two styles f_a and f_b based on their font features, in the following order: font size (desc.), presence of a font face (bold or italic), a font color different from default, and the number of occurrences of the style (asc.). Then, BODY_TEXT_STYLE is used to identify the body text style as the formatting style with the highest number of occurrences throughout the textbook: $f \mid o(f) \text{ is } \max\{o(x) : \forall x \in F\}$, where $o(\cdot)$ denotes the function that returns the number of word-occurrences of a formatting style f . Finally, LABEL_OF_STYLES is used to label each style according to possible roles: heading, subheading, body text, caption, formula, footnote, etc. Currently, each style that is higher than the body text in the sorted set of styles is classified as heading or subheading. This rule is being extended to recognize the other possible roles.

Some labels can be misidentified due to special styles used for special texts: quotes, formulae, remarks, etc. Correcting labels is done after the entries from the Table of Contents have been identified in the next stage (🔗 Section 3.3.4). LABEL_OF_STYLES_CROSS_CHECK finds for each entry e in the ToC the style for the line l_x that matches the title of e in its corresponding beginning page. The exact title line is found using MATCHING_LINES. Finally, styles in the sorted set of styles that do not correspond to the found styles using the ToC entries are deleted.

Labels Assignment

In this step, TEXT_FRAGMENTS groups adjacent lines (l_x, l_{x+1}, \dots) with the same formatting style f to form labeled text fragments. As a result, a new textual element t , as a text fragment, is introduced. The list $T_y = (t_{1y}, t_{2y}, \dots, t_{ny})$ corresponds to a page p_y formed by text fragments. Now, P is formed by pages, fragments, lines, and words.

3.3.4 Logical Element Identification

The third large stage of the workflow is to recognize all different logical elements within a textbook.

Auxiliary Elements Filtering

First, REPEATED_LINES is used to detect (and then remove) header and footer lines across pages. A sample of pages $P_s = (p_a, p_b, \dots, p_m) \mid P_s \subset P$ is selected. If the first and/or last line(s) are identical across P_s , they are removed in all pages p .

Copyright entries undergo similar treatment. They often appear at the start of chapters and are identified and removed using COPYRIGHT_TEXT. This rule compares each word in the last five lines from the first page of each chapter against a predefined list of commonly-used copyright symbols (e.g., ©, DOI, ISBN). If there are at least two matches, the fragment is removed. Since this rule uses the first page of each chapter, the workflow actually needs to postpone its application until the sections have been identified in the last step of the current stage (🔗 Step Section Identification).

The final type of auxiliary page elements that have been processed are the page numbers. The workflow does not filter them out as they provide important information for cross-referencing and navigation. Yet, they often require correction. The *page numbers* (PN) printed on pages have to be matched to the *ordered pages numbers* (OPN) within P . These two numbers are usually different since textbooks start with a cover, non-numbered front matter, blank pages, etc. Two rules are used during the matching: POSITION_OF_PAGE_NUMBERS and MISSING_PAGE_NUMBERS. The former scans the top and bottom lines of the pages; if a line contains a number and the following pages follow the numbering sequence, the position of the page number is stored. Then, for each $p \in P$, the corresponding page number is retrieved. The latter rule checks all the pages in P in the descending order, and when there is a page without a page number, it assigns to the page the previous page number minus one, if it is not already assigned to another page. The mapping $N : PN \rightarrow OPN$ is added to M . Now, $M = (P, N)$.

Table of Contents Identification

The ToC section provides organizational and browsing information of textbooks. A ToC indicates not only the hierarchical arrangement of chapters and subchapters in the text, but also provides an overview of the topics and subtopics of the textbooks. This knowledge that was defined carefully by the author(s) provides structural and domain-specific information relevant for the knowledge model of the textbook. In total, ten rules are used in this stage.

First, BEGINNING_OF_TOC is used to detect the first page of the ToC section by comparing heading text fragments from a list of pages P_{start} to a list of predefined words for the language of the textbook. P_{start} contains the first 50 pages in P . Previous work [291, 293] used the first 20 pages of the book for this step, but while developing the approach, textbooks were found where that margin is not enough. The first ToC page is added to P_{toc} , which is the list containing all pages of the ToC. Next, the remaining ToC pages are identified and added to P_{toc} with TOC_PAGE_LAYOUT. This rule detects pages that follow the ToC layout of entries and stops when a head-

ing fragment is reached. Then, for each $p \in P_{toc}$, `PAGE_MARGIN_CORRECTION` checks and aligns the left page margin for the subsequent rules to correctly compute indentations of ToC entries and infer ToC section hierarchy. All lines from the pages in P_{toc} are extracted and added to a list ToC .

`MULTILINE_ENTRY` is used to concatenate multiple lines $l_x \in ToC$ that belong to the same ToC entry e . A line is considered incomplete if it does not end with a number from PN that is placed in the same position as other page numbers in the ToC. An incomplete line should be merged with consequent lines until a complete line is reached. After this rule $ToC = (e_0, e_1, \dots, e_n)$.

There are three special ToC entries. The beginning of the ToC, can contain entries for introductory sections (e.g., *preface*) that are non-content chapters of the textbook. These sections usually use roman numbers; therefore, `INTRODUCTORY_ENTRY` uses a regular expression for detecting those entries. Some ToC entries include a list of authors, these entries are detected with `AUTHOR_ENTRY`, which uses a named-entity recognition algorithm to detect if the majority of the words in a line correspond to names of persons. Introductory and Author entries are removed from ToC . `PART_ENTRY` detects entries that correspond to the title of a part, instead of a chapter, if the line spacing before and after an entry is bigger than the one used for the majority of the top-level entries.

Similar to the approach of Wu et al. [293], the rule `TOC_TYPE` is used to classify the structure of the ToC into one of the three possible cases: *flat* (each entry is on the same level of hierarchy), *flat-ordered* (flat with ordered chapter and subchapter numbers), and *indented* (the ordering is given by different levels of indentation); formally:

$$\begin{aligned} \text{if } \forall e \in ToC, s(e) = x \text{ and } n(e) = 0, \text{ type} &\mapsto \textit{flat}; \\ \text{if } \forall e \in ToC, s(e) = x \text{ and } n(e) \neq 0, \text{ type} &\mapsto \textit{flat-ordered}; \\ \text{otherwise, type} &\mapsto \textit{indented}. \end{aligned} \quad (3.1)$$

The function $s(\cdot)$ returns the starting X-coordinate of a entry e or a line l_x . The function $n(\cdot)$ returns the number of elements in the enumeration that represents the ordered chapter or subchapter number (e.g., "1.2.1" has 3 elements), or 0 if such number is not present in the entry.

For each specific type of ToC, `HIERARCHY_OF_ENTRY` is applied to each entry e to build a hierarchy of the sections of the textbook (parts, chapters, and subchapters). The function $h(\cdot)$ returns the hierarchy number for any e , initially $\forall e \in ToC, h(e) = 0$. For a *flat* ToC, the rule is: $\forall e \in ToC, h(e) \mapsto 1$. For a *flat-ordered* ToC: $\forall e \in ToC, h(e) \mapsto n(e)$. Finally, for an *indented* ToC, $\forall e \in ToC$:

$$\begin{aligned} \text{if } s(e) = \min\{s(x) : \forall x \in ToC\}, h(e) &\mapsto 1; \\ \text{if } s(e) = s(pr(e)), h(e) &\mapsto h(pr(e)); \\ \text{if } s(e) > s(pr(e)), h(e) &\mapsto h(pr(e)) + 1; \\ \text{if } s(e) < s(pr(e)), \exists x \in ToC \text{ and } h(x) \neq 0 \text{ and } s(e) = s(x), h(e) &\mapsto h(x). \\ pr(e_x) &= e_{x-1}. \end{aligned} \quad (3.2)$$

ToC entries contain the starting page number of a section, but its end page is not 100% known. `END_OF_SECTION` is used to find the ending page number of each entry e_x by taking the next entry e_{x+1} and locating its title in its starting page (using `MATCHING_LINES`). If the title is the first line of the page, $end(e_x) \mapsto (start(e_{x+1}) - 1)$, otherwise $end(e_x) \mapsto (start(e_{x+1}))$. $start(\cdot)$ returns the starting page number of a section according to ToC , and $end(\cdot)$ returns the learned ending page number of a section. The final hierarchy of entries ToC , is added to M . Now, $M = (P, N, ToC)$. Figure 3.2.a shows different elements in the ToC that are recognized by the rules.

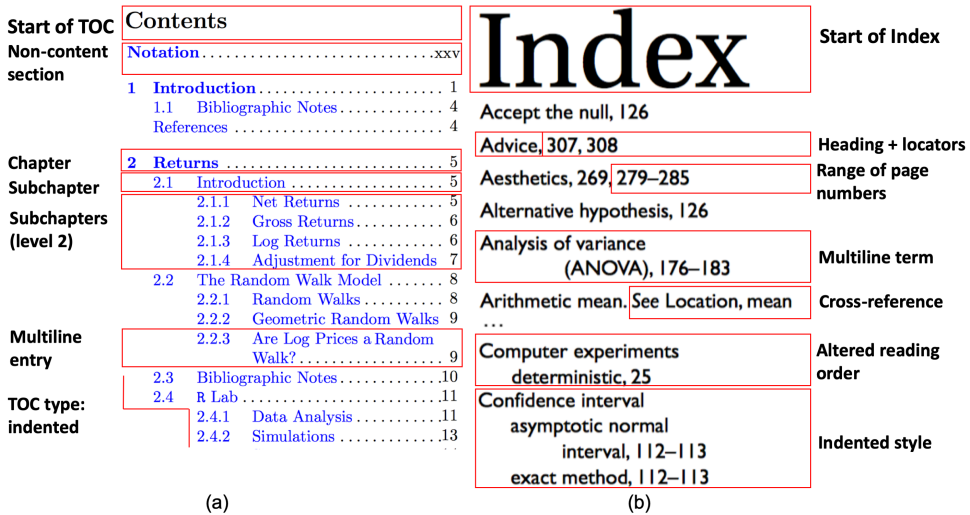


Figure 3.2: Examples of elements identified in (a) ToC and (b) Index sections.

Index Identification

The index section keeps track of the introduction point of every relevant term within a textbook. A useful index is essentially a reference model produced by a domain expert according to a predefined set of rules. Each entry in this model is provided with one or more links to the pages within a textbook—pages that either introduce corresponding terms or elaborate them. This consistency of terms enables to create connections within the textbook and among different textbooks in the same domain, which makes this section important for the knowledge model of the book. Index entries are identified using 12 rules.

First, `BEGINNING_OF_INDEX` uses a list of predefined words for the language of the textbook to detect the beginning of the section in the already recognized ToC entries. If the corresponding entry is not found, the heading text fragments from a list P_{end} , which contains the last 20 pages of the textbook, are compared against the list of predefined words. Then, the other pages of the index are identified by applying `INDEX_PAGE_LAYOUT` to the pages following the first index page. This rule detects if the majority of the lines end with a valid page reference, and no heading text fragment is found in the page. Identified pages p with an index layout are added to P_i , which is the list that contains all the index pages. Then, for each index page $p \in$

P_i , MULTICOLUMN_LAYOUT is applied to group lines l_x from the page into columns. Since index terms can be nested and the text alignment is often different across columns and pages, COLUMN_MARGIN_CORRECTION aligns the starting X-coordinate of all the columns. This rule first learns the most left X-coordinate for each column; then, using simple majority, the most used X-coordinate that represents the first-level index entry is selected for each column of even and odd pages in P_i . Finally, each column is aligned with the first column of the first index page. After this rule, all the lines from all columns are extracted and added to a list called I .

The next part of the step involves rules to clean and detect each individual index term i from all the lines $l_x \in I$. First, ALPHABET_LETTER detects lines with only one letter using a regular expression, which indicates the titles for the alphabetization of index entries. The detected lines are removed from I . Then, DEHYPHENATION is used to merge separated words. LOCATOR_ELEMENT, TERM_DELIMITER, and CROSS_REFERENCE are used together to detect the parts of an index term: the heading or subheading, the locators, and cross-references. The heading and subheading are nouns or noun phrases that represent one unit of knowledge relevant to the textbook⁷, the locators are the associated page references, and cross-references are references to other index terms. LOCATOR_ELEMENT identifies five different types of page references: single roman number (e.g., V), range of roman numbers (e.g., V-VIII), single page number (e.g., 45), range of page numbers (e.g., 15-17), and endnotes (e.g., 25N3). Each type is first identified using a dedicated regular expression, and then normalized to a sequence of page numbers: roman number references are discarded, all the page numbers from the ranges are generated, and note numbers are discarded. TERM_DELIMITER identifies the delimiter used in the index to separate the heading/subheading from the locators, which is usually a comma or a blank space, by choosing the most used character that separates non-locator elements from locator elements. CROSS_REFERENCE identifies the words used for cross references using a list of predefined words for the language of the textbook (usually, "see" and "see also" in English). Additionally, the rule checks that the identified cross-reference words appear after a locator or have a different style from the (sub)heading of the term to avoid mismatches.

In order to process all the lines that belong to I and create individual index terms i , multiple line index entries and grouped index terms should be identified. MULTILINE_TERM indicates if two lines l_x and l_{x+1} should be concatenated because they represent a single term according to the following cases: 1) if l_{x+1} only has locator elements; 2) if l_x does not have any locator element, l_{x+1} has a least one locator element, and l_x does not form a nested group with more lines. The rule is applied multiple times until no new lines are merged to account for big unique index entries.

Often, index terms are grouped into nested hierarchies. Sometimes authors model broad index terms by first putting an index term with or without locators and then adding related terms with an indentation in the next lines. Nested hierarchies are identified using four rules. First, groups of index terms are formed using FLUSH_AND_HANG: for each $l_x \in I$, it is grouped together with the following lines (l_{x+1}, l_{x+2}, \dots) if $s(l_{x+i}) > s(l_x)$. Sometimes if a nested index term starts at a page

⁷Name of a person, a place, an object, or an abstraction.

and continues on the following, the first term is repeated in the second page with the added word "cont." or "continued", in that case, two different groups are created with the previous rule. `CONTINUED_TERMS` is used to detect and merge such groups. Then, groups are resolved to individual index terms using two rules: `RUN-IN_STYLE` and `INDENTED_STYLE`. The first rule checks if one line contains several complete index terms (heading + locators) separated by semicolons. If a run-in entry is detected, it is separated into parts using the character ";" as a breaking point, and then the first term is concatenated separately to each other term to form individual index terms. `INDENTED_STYLE` concatenates the first index term in a group with the index terms resulting from applying `RUN-IN_STYLE` or `INDENTED_STYLE` recursively to each child term in the group.

The final list of index entries, I , is added to M . Now, $M = (P, N, ToC, I)$. Figure 3.2.b shows the different elements in the Index section that are recognized using the rules.

Section Identification

The content of each section is identified and extracted in both PDF and TXT formats in this step. For the PDF version, using the starting and ending page for each section is enough to create a subset of the original PDF. This partitioning means that the PDF of a chapter or subchapter could contain part of the previous or following section if they share a page. In contrast, the TXT segment of each section only includes the text for that chapter or subchapter. Here, `SECTION_CONTENT` uses the information from ToC and P , along with `MATCHING_LINES` to find the precise section boundaries at the starting and ending page for each section. Then, `MULTICOLUMN_LAYOUT` and `DEHYPHENATION` are used to produce an accurate text version of each section.

The PDF segments allow to share specific parts of the textbook easily, and the textual content enables more accurate indexing to enable additional services.

After the textual content of each section is extracted, `CORRESPONDING_SECTION` links index terms to sections containing pages of their occurrences. If a target page belongs to one possible section, the linking is straightforward. If the page is shared by two or more possible sections, a search algorithm is applied to find the term in the text corresponding to a particular section. Each entry $i \in I$ is updated with its corresponding sections.

The list PDF contains a mapping between each entry $e \in ToC$ to its corresponding PDF segment. The list TXT contains a section entry s_x for each $e_x \in ToC$. Each $s \in TXT$ describes its precise starting and ending line numbers (section boundaries), title, and textual content. PDF and TXT are added to M . Now, $M = (P, N, ToC, I, PDF, TXT)$.

3.3.5 Model Construction

After all the structural information has been extracted, the final stage is to create a representation of a textbook.

TEI Model Construction

The Text Encoding Initiative (TEI)⁸ is used to formally express the content and the structure of textbooks. TEI Guidelines provide a standard for digital representation of texts. TEI is widely used by digital libraries, museums, publishers, and researchers to represent various kinds of texts for a variety of purposes. The P5 Guidelines are used to express the information collected in model M . In this step, the elements from M are mapped to TEI elements. The TEI model groups the information from M into three categories that represent the information extracted from the textbook: Structure, Content, and Domain Knowledge. Table 3.1 presents the mapping between the extracted information and the used TEI elements. The choices for the TEI elements share similarities with a previous proposal of TEI elements for textbook research [255].

3.4 Evaluation

Two evaluation experiments have been conducted. The first one tested the accuracy of the outcomes produced by the approach that are used to construct the textbook model. The second experiment examined how the obtained model can support one of its possible knowledge-driven applications: linking relevant sections across textbooks within the same domain.

3.4.1 Accuracy of the Approach

The goal of the first experiment has been to find out if the approach can correctly identify and extract textual and structural information as well as domain terminology from PDF-based textbooks as the accuracy of this information directly affects the quality of resulting models. In order to do so, the information extracted from the PDF versions of textbooks has been verified against the information obtained from HTML tag elements and CSS classes of the EPUB versions of the same textbooks. Parsing an XML-based EPUB format is a straightforward procedure with a guaranteed outcome; hence, it is used as the ground truth. The author hypothesizes that if the information obtained from the two versions of a textbook matches, that means the approach processes PDF correctly. First, the results of the approach have been compared against two popular PDF processing tools to demonstrate the added value of the rule-based workflow when extracting raw text. In addition, it has been verified how well the approach elicits structural information focusing on ToC entries and index terms.

Procedure

SpringerLink has been used as the source repository for the test set as it provides a large number of high-quality textbooks in several domains. *Statistics* has been chosen as the main domain. A set of 40 textbooks on university-level Statistics written in English, available in both PDF and EPUB versions, and containing Tables of Content and Index sections have been selected. Additionally, to validate the approach

⁸<https://tei-c.org/>

Table 3.1: Map between the internal textbook model and the TEI textbook model.

Category	Textbook component	TEI elements	TEI attributes / values	Notes
Structure	ToC section (ToC)	$\langle div \rangle$ (text division)	@type="contents"	Content of $\langle ref \rangle$ is the page number, target points to its content section
	ToC entries ($e_x \in ToC$)	$\langle list \rangle + \langle item \rangle + \langle ref \rangle$ (reference)	$\langle ref \ @target="segment id" \rangle$	
Content	Part, chapter, or subchapter ($ToC + TXT$)	$\langle div \rangle$ (text division)	@type="part chapter section" @id="[segment id]"	
	Title of part, chapter, or subchapter ($s_x \in TXT$)	$\langle head \rangle$ (heading)		Heading/subheading
	Pages ($p_z \in P$) + page numbers (N)	$\langle pb \rangle$ (page beginning)	@n="[page number]"	Body text fragments
	Fragments ($t_y \in T_z$)	$\langle ab \rangle$ (anonymous block)		
	Lines ($l_x \in L_y$)	$\langle lb \rangle$ (line beginning)		
Domain Knowledge	Words ($w_u \in W_x$)	$\langle w \rangle$ (word)		
	Index section (I)	$\langle div \rangle$ (text division)	@type="index"	Content of $\langle ref \rangle$ is a page reference, target points to its content section.
	Index terms ($i_x \in I$)	$\langle list \rangle + \langle item \rangle + \langle label \rangle + \langle ref \rangle$ (reference)	$\langle ref \ @target="segment id" \rangle$	Label is the reading label

in other domains, five textbooks have been added to the test set for each of the following domains: *computer science*, *history*, and *literature*⁹.

This experiment has focused on three different outcomes from the first three stages of the approach (the last stage is the construction of the textbook knowledge model itself). To evaluate the quality of text extraction stage, its results have been compared with the outcomes of two state-of-the-art PDF text extractors: PDFBox and PdfAct (formerly known as Icecite). PDFBox is the underlying tool used for text extraction in the proposed approach, but without access to the rule-based inference. PdfAct has been chosen, because it supports several advanced text processing features such as identification of semantic roles and merging of hyphenated words; hence, its functionality is conceptually comparable to this chapter's approach. Additionally, in the mentioned evaluation of PDF text extractors [26], it yielded satisfactory results on all evaluation criteria. PDF versions of the entire corpus of textbooks from 4 domains were processed by all three tools. *Google's Diff Match Patch library*¹⁰ was used to synchronize and compare the extracted texts with the ground truth generated using the EPUB versions of the textbooks. Table 3.2 shows the results for the text extraction evaluation. For each tool, the number of characters in the PDF documents and the percentage of characters that match in both versions are averaged over all the evaluated textbooks.

Evaluation of structural information has been done by comparing the identified ToC entries against the ToC sections from the ground truth. Following the evaluation procedure used in the ICDAR'2009 and ICDAR'2013 competitions [78, 79], each entry is classified as correct if the right heading, hierarchy level, and page number match the ground truth. Standard precision and recall [214, chap. 9] are reported over the total number of evaluated elements.

Finally, the extracted domain terms have been evaluated by comparing each index term against the Index section from the ground truth. For each individual index term, the right label and hierarchy have been compared (page numbers have been ignored since EPUB textbooks do not have fixed page numbers). This evaluation has also used standard precision and recall over the total number of evaluated elements as metrics. Table 3.3 shows the results for the structural and domain terms evaluations.

Analysis

The results of the text extraction evaluation show that the proposed rule-based system demonstrates the best accuracy among the three evaluated tools, followed by PDFBox and then PdfAct (Table 3.2). The results do not reach the 100% mark because of the four main reasons. First, front matter and back matter of the PDF and EPUB versions are often very different. Second, some textual characters extracted from PDF are represented only visually but not textually in EPUB and therefore not extracted (e.g., bullet points). Third, many symbols in formulas, as well as textual labels in charts and tables are represented as images in EPUB but as text in PDF. Finally, some fonts in PDF textbooks (especially, *Type-3 fonts*) provide incorrect

⁹The same selection criteria have been applied: English language, PDF and EPUB versions, Table of Contents and Index section.

¹⁰<https://github.com/google/diff-match-patch>

Table 3.2: Text Extraction.

Tool / Domain		Statistics	Computer Science	History	Literature
EPUB	Chars (#)	641697	373615	662300	571277
Proposed approach	Chars (#)	730060	374985	660493	568561
	Matched (%)	82.09	96.61	98.07	98.64
PDFBox	Chars (#)	737015	410238	671693	578121
	Matched (%)	79.93	85.42	96.50	97.03
PdfAct	Chars (#)	727502	407373	661503	565463
	Matched (%)	74.04	81.34	91.51	89.89

Table 3.3: Extraction of ToC entries and domain terms (index).

Model / Domain		Statistics	Computer Science	History	Literature
ToC	#	7968	811	216	113
	Precision (%)	99.70	100	100	100
	Recall (%)	99.70	100	100	100
Index	#	19779	1661	2567	1800
	Precision (%)	99.78	97.77	97.78	98.94
	Recall (%)	99.80	98.00	96.66	98.07

mappings for glyph and Unicode characters. The last two cases have been especially prominent for the statistics textbooks, where around 20% of characters extracted from PDF textbooks are missing in their EPUB versions. An additional effect of the rules that improve textual extraction (e.g., sorting the characters, and merging of hyphenated words), along with the rules for recognition of page elements (e.g., headers and pager numbers) is a cleaner textual version of the textbook, as seen when the proposed approach is compared against the out-of-the-box PDFBox tool that lacks these features. PdfAct was designed for scientific articles, and therefore, it does not handle textbooks as good as the two other methods.

When it comes to the identification of structural elements, precision and recall values are high across all domains (Table 3.3: ToC). Results for the detection of ToC entries show that the number of recognized entries is always correct (precision and recall are the same), but in a few cases, the text or the hierarchy is wrong. After manual inspection, it was observed that some entries were misplaced in the ToC hierarchy (e.g., third-level chapter instead of second-level) because of the wrongly recognized mathematical characters (e.g., $\sqrt{\quad}$). These cases are related to the text extraction rules, where improvements will be made. Finally, for domain terms identification both precision and recall are very high as well (Table 3.3: Index). Several terms have been recognized incorrectly because of special cases that the defined

rules could not always handle: when the delimiter between heading and locators are inside quotes (e.g., "Castel," 192), and when a range of numbers do not follow a complete ordered sequence (e.g., 677-671 or 701-5). A few problems were due to the dehyphenation of words (e.g., *t-* was merged with *distribution*). As a result of these findings, corresponding rules will be modified.

Overall, this experiment has demonstrated that the proposed rule-based approach extracts textual, structural, and domain term information from the PDF textbooks with high precision and recall across different domains.

3.4.2 Knowledge Model Application: Linking of Sections

The goal of this experiment has been to test one of the possible knowledge-driven applications of extracted models. Specifically, the knowledge models of two textbooks have been used to cross-link relevant sections between them. Additionally, the results from the linking model have been compared to two baselines. The author believed that the use of the extracted domain knowledge of the textbooks will produce accurate matches between the sections of textbooks.

Procedure

This evaluation has followed a procedure similar to the one presented by Guerra et al. [117]. Two introductory statistics textbooks have been used: BOOK#1 [282] and BOOK#2 [72]. Three experts from Utrecht University have been employed to produce the ground truth for this experiment by manually mapping chapters, sections and subsections of BOOK#1 to the parts of BOOK#2. No restrictions have been imposed on the mapping granularity or coherence. The experts could cross-link sections on any levels of textbooks' ToCs, could produce n-to-m mappings and could specify the strength of the mapping relations.

A term-based knowledge model of each textbook has been extracted by the presented approach. Additionally, they have been automatically mapped to the ISI Multilingual Glossary of Statistical Terms (ISI Glossary)¹¹. The ISI Glossary translates more than 3500 statistical terms (and synonyms) into 31 languages¹². As a result, the two original models have formed a unified reference set of 1611 terms. This set was applied to build a term-based VSM [238] of all (sub)chapters and (sub)sections of both books. The sections have been annotated by the terms according to the knowledge models extracted from the textbooks' indices. The inner product of these annotations has been used to compute similarity between all sections of BOOK#1, and sections of BOOK#2.

To evaluate the effectiveness of the linking model, average NDCG@1, @3, and @5 scores have been used. Normalized Discounted Cumulative Gain (NDCG)[139] is a measure of the quality of ranking documents by relevance. NDCG@1 measures the effectiveness of retrieving the most relevant document, while @3 and @5 measure the capability of the retrieval system to find the first three and five most relevant documents, respectively. The linking produced by the experts has been used as the ground truth for the NDCG measures. Additionally, two baselines have been used for

¹¹<http://isi.cbs.nl/glossary/>

¹²It was created by The International Statistical Institute which recognized the need for an affordable multilingual glossary of statistical terms.

comparison: the standard *TFIDF* model [239] and the *LDA* model [31] built based on BOOK#1 using 2000 iterations, 158 topics (the number of sections in BOOK#1), and $\alpha = 50$. The Hellinger distance has been used as the similarity measure for LDA. Both baselines have used the textual content of each part of the textbooks with basic preprocessing (lowercase, stop-words, and stemming). Additionally, an ensemble model has been computed using the two baselines where each method contributes equally to the final score.

NDCG@1, @3, and @5 mean values and standard deviations for all models are presented in Table 3.4.

Table 3.4: Semantic Linking of Textbook Sections.

Metric / Model		TFIDF	LDA	TFIDF+LDA	TERMS
NDCG@1	M	.428	.458	.494	.721
	SD	.487	.490	.493	.431
	t	5.56	5.03	4.21	-
	df	123	123	123	-
	Sig.	<.001	<.001	<.001	-
NDCG@3	M	.619	.710	.727	.795
	SD	.430	.403	.385	.344
	t	4.43	2.20	2.00	-
	df	123	123	123	-
	Sig.	<.001	.177	.287	-
NDCG@5	M	.679	.774	.774	.834
	SD	.371	.322	.322	.290
	t	4.69	2.00	2.14	-
	df	123	123	123	-
	Sig.	<.001	.287	.206	-

Analysis

The results show that the proposed model (TERMS) consistently outperforms all baselines. The results obtained from the baseline models are logical and comparable to previous research [117]: TFIDF model has the lowest accuracy, followed by LDA, followed by the ensemble model. The difference between the TERMS model and the baselines is the highest for NDCG@1. The semantic information placed by the authors of textbooks in the index sections and extracted by the proposed approach helps the TERMS model find 72.1% of best possible matches between the textbook sections. As the number of potential matches increases the difference between NDCG scores diminishes due to the ceiling effect. When providing the ground truth, experts did not always agree, yet the number of matches per section rarely went up to five. At the same time, the TERMS model does not gain substantial benefits from relaxing the matching requirements as it already ranks the most relevant sections higher than LDA and TFIDF. A range of pairwise t-test has been conducted to check if TERMS significantly outperforms the baselines across the BOOK#1 sec-

tions. Table 3.4 presents the results including Bonferroni-adjusted p-values.

3.5 Conclusions and Future Work

This chapter presents the implementation details and the evaluation of a novel approach for automated extraction of knowledge models from textbooks. Results of the first evaluation experiment show that the proposed approach is capable of processing PDF textbooks with high accuracy—structure, content, and domain terms are correctly extracted. The second experiment demonstrates the added value of the extracted knowledge models by using them to link sections across textbooks within the same domain. This approach is the foundation for Intextbooks: a system capable of transforming PDF textbooks into intelligent educational resources (👉 Chapter 7, Section 7.3).

It is important to underline, that the models extracted with the help of this approach are not guaranteed to provide high-quality representation of domain knowledge. For example, they can potentially suffer from low coverage. To combat some of the problems, models extracted from individual textbooks within the same domain can be integrated with each other and with external linked datasets. Preliminary results in this direction are promising (👉 Chapter 4).

As a remark, while the core approach is largely agnostic to the language of a textbook, some of the rules use a predefined list of words to detect various textbook elements. Currently, lists for textbooks written in English, German, French, Spanish, and Dutch have been created. Supporting additional languages would require creating corresponding lists. An interesting application that can benefit from the multilingual support is the Interlingua platform where students can study textbooks in a foreign language while getting on-demand access to relevant reading material in their mother tongue (👉 Chapter 7, Section 7.2).

The future work is planned in two directions, to extend the approach further and improve the extraction of models from a broader range of textbooks and domains. Ultimately, the extracted models can provide a semantic skeleton for a range of possible applications, including reasoning and inference, filtering and retrieval, navigation and assessment. Besides linking, for example, in the presence of a such a model, student's reading behavior can be not only traced but also interpreted in terms of domain knowledge. As a result, an interface providing adaptive reading support can be implemented navigating students towards the most relevant/necessary fragments of a textbook.

CHAPTER 4

Expanding the Web of Knowledge: One Textbook at a Time

Abstract Textbooks are educational documents created, structured, and formatted to facilitate understanding. Most digital textbooks are released as mere digital copies of their printed counterparts. In this chapter, the extracted knowledge models are enriched with additional links (both internal and external). The textbooks essentially become hypertext documents where individual pages are annotated with important concepts in the domain. This chapter also shows that extracted models can be automatically connected to the Linked Open Data cloud, which helps further facilitate access, discovery, enrichment, and adaptation of textbook content. Integrating multiple textbooks from the same domain increases the coverage of the composite model while keeping its accuracy relatively high. The overall results of the evaluation show that the proposed approach can generate models of good quality and is applicable across multiple domains.


This chapter is based on the following publications:

Alpizar-Chacon, I. & Sosnovsky, S., “Expanding the web of knowledge: one textbook at a time”, in: *Proceedings of the 30th on Hypertext and Social Media*, HT’19, Hof, Germany: ACM, 2019. **ACM HT 19 Ted Nelson Newcomer Award.**

Alpizar-Chacon, I. & Sosnovsky, S., “Knowledge models from pdf textbooks”, *New Review of Hypermedia and Multimedia*, vol. 27, no. 1-2, 2021, pp. 128–176.

4.1 Introduction

Nowadays, a lot of high-quality expository content is accessed online. Textbooks, tutorials, manuals, etc. are available on the WWW for many subjects. Most of them are carefully-written and well-formatted collections of thematic texts that are selected, ordered and structured in a way that facilitates understanding. Textbooks in particular tend to be logical and consistent in their structural organization and formatting elements. These components of a good textbook design are vehicles for improving structure awareness in readers, which is an important factor for helping text comprehension and recall [122]. Unfortunately, digital textbooks are rarely published as hypertext documents. Even fewer such textbooks are provided with additional tools, components or models that facilitate knowledge discovery within a book, or linking to external knowledge models and/or relevant resources outside of it. Considering how much digital content is available online on virtually any topic, readers of these textbooks could greatly benefit from some level of support when they try to search for or navigate to information on a certain topic.

The previous chapter ( Chapter 3) presented an approach towards automated extraction of machine-readable domain models from textbooks taking into account their formatting rules and internal structure. The presented approach focused on PDF as the most common textbook representation format; however, the overall approach is also applicable to other formats that are more explicit and coherent in their structural specifications than PDF. This chapter takes the next step and demonstrates how these models can automatically link textbooks from the same domain and integrate textbooks with a global reference model, such as DBpedia. As a result, textbooks can potentially become ubiquitous sources of knowledge for a growing Linked Open Data (LOD) initiative ¹. Textbooks themselves also transform into collections of documents linked to this LOD cloud and become available for a rich variety of services that can facilitate their discovery, enrichment, adaptation, etc.

Additionally, to combat the potential low coverage of the domain in the generated knowledge models, glossaries extracted from individual textbooks can be integrated with each other and linked to external models. This study focuses on one of the most popular such models—DBpedia [30]. It is a machine-understandable knowledge base automatically-extracted from Wikipedia. Currently, DBpedia contains 4.58 million entities and is available in 125 languages. Its information is stored as RDF triples that can be queried using a SPARQL endpoint. The entire model can be also downloaded as a database dump.

The main contributions of this chapter are: (1) an approach for the discovery of entities belonging to specific domains in a Knowledge Base using index terms extracted from a textbook; and (2) evaluation of the quality of the linking strategy and the discovery of entities in DBpedia.

The remainder of this chapter is organized as follows. Section 4.2 presents related work. Section 4.3 provides the details of the proposed approach. Evaluation is described and discussed in Section 4.4. Finally, Section 4.5 outlines conclusions and future work.

¹<https://lod-cloud.net/>

4.2 Related Work

4.2.1 Knowledge Discovery and Terminology Extraction

DBpedia and other KBs have been used before for exploratory knowledge search. Semantic Wonder Cloud [199] is a graphical tool where the user selects an initial DBpedia entity, and then, the tool displays the ten most similar entities using a hybrid ranking algorithm. The user can continue the exploration by selecting one of the entities to get a new set of related entities. Discovery Hub [179] is an exploratory search engine where the user selects one or several topics of interest, and then, the system selects and ranks on-the-fly a meaningful subset of entities using a spreading activation algorithm and a sampling technique. Medelyan et al. [189] use Wikipedia articles as topics and their titles as controlled terms to discover topics in documents.

Terminology Extraction tasks deal with automated identification of core vocabulary of a specialized domain in un- and semi-structured corpora. Dwarakanath et al. [82] present a two-step method for the automatic extraction of glossary terms from software requirements documented in natural language. Lopes et al. [171] use three different methods to extract compound terms from a text corpus of the domain of pediatrics.

The proposed approach in this chapter is not guided by one initial entity in particular, but instead, it uses a domain-specific glossary extracted from the index sections of textbooks to discover the entities in DBpedia that belong to the same target domain.

4.2.2 Linking Resources to Entities

The process of identifying the true "sense" (meaning) of a resource or linking it to a formally defined entity has been researched in closely related fields. Named Entity Recognition (NER) is an approach to identify and classify named entities in free text [121]. Word Sense Disambiguation (WSD) is the task of choosing the right sense for a word in a context using an exhaustive dictionary [51, 204]. Entity Linking (EL) and Named Entity Disambiguation (NED) are sometimes used indistinctly, where EL refers first to identifying entities and then linking them to an entity in a KB, and NED focuses on the potential ambiguity among several possible candidates in a KB [51]. Typically, lexical ontologies, such as WordNet², have been used in WSD [6, 208], and global knowledge bases such as DBpedia in EL and NED [155, 197]. This chapter focuses on NED.

Different approaches and tools to solve NED tasks have been proposed. DBpedia Spotlight [192] receives a text, detects the entities, and applies a VSM using context around the entities in DBpedia and the input text to disambiguate possible candidates for the entities. Babelfly [204] computes semantic signatures for concepts using Wikipedia, and then it links entities to the concepts based on a high-coherence densest subgraph algorithm. TAGME [97] uses a collective agreement between a candidate entity and the possible candidates for other entities in the text for disambiguation. KORE [130] uses keyphrase overlap relatedness to relate entities in

²<https://wordnet.princeton.edu/>

the text to pre-extracted entities from Wikipedia. Other approaches use a notion of exclusivity-base relatedness to measure how similar two nodes in a graph are [111], and a common subsumers algorithm between ambiguous entities and close entities that are unambiguous [133] for disambiguation.

The proposed disambiguation strategy in this chapter differs from others in several aspects. First, it uses the fact that textbooks belong to a specific domain by creating a seed set of DBpedia entities for context, if one is not provided. The set is created using a DBpedia category that matches the target domain. This category is the only manual input data that the algorithm requires, in comparison to keywords [232] and seed entities [198] in other approaches. Second, the list of keywords from textbooks are in most cases abstract concepts and not concrete PLO (Person, Location, and Organization) entities, so the use of individual categories or special formatting patterns for identifying those entities cannot be used [35, 70, 155]. Finally, relying on prior probabilities to get the most popular [123, 197] or authoritative [275] entities is not useful in the context of textbooks because they use domain-specific terms and not general entities. The disambiguation algorithm uses a DBpedia category to construct a seed set of entities, then it extracts the abstracts from the seed set to be used as context information, and finally, it uses a similarity measure based on the cosine coefficient to disambiguate between similar candidates for each index term.

4.2.3 Open Knowledge Bases

Finally, it is important to mention that DBpedia is not the only openly available global knowledge base that can be used to retrieve entities in the Semantic Web [94]. Wikidata [279] started in 2012, and is an open KB that can be read and edited by both humans and computer agents. It stores facts extracted from the structured data of Wikipedia, Wiktionary, and other projects of the Wikimedia foundation. Currently, Wikidata contains more than 63 million items. It provides dumps for its database in different standard formats, and also offers a SPARQL endpoint for direct querying. YAGO³ has been developed since 2007. It has imported information from Wikipedia, WordNet, and GeoNames⁴. YAGO stores more than 10 million entities derived from about 120 million facts. Its latest version, YAGO3, is available for download. KBpedia⁵ is another project combining knowledge from several public knowledge bases: Wikipedia, Wikidata, schema.org, DBpedia, GeoNames, OpenCyc⁶, and UMBEL⁷. It includes 55 thousand reference concepts, links to about 32 million entities, and 5 thousand relations and properties. KBpedia provides an online search tool and it is also available for download. Even though the proposed approach uses DBpedia, it could be adapted to work with other knowledge bases.

³<https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/downloads/>

⁴<http://www.geonames.org/>

⁵<http://kbpedia.org/>

⁶<https://www.cyc.com/opencyc/>

⁷<http://umbel.org/>

4.3 Approach

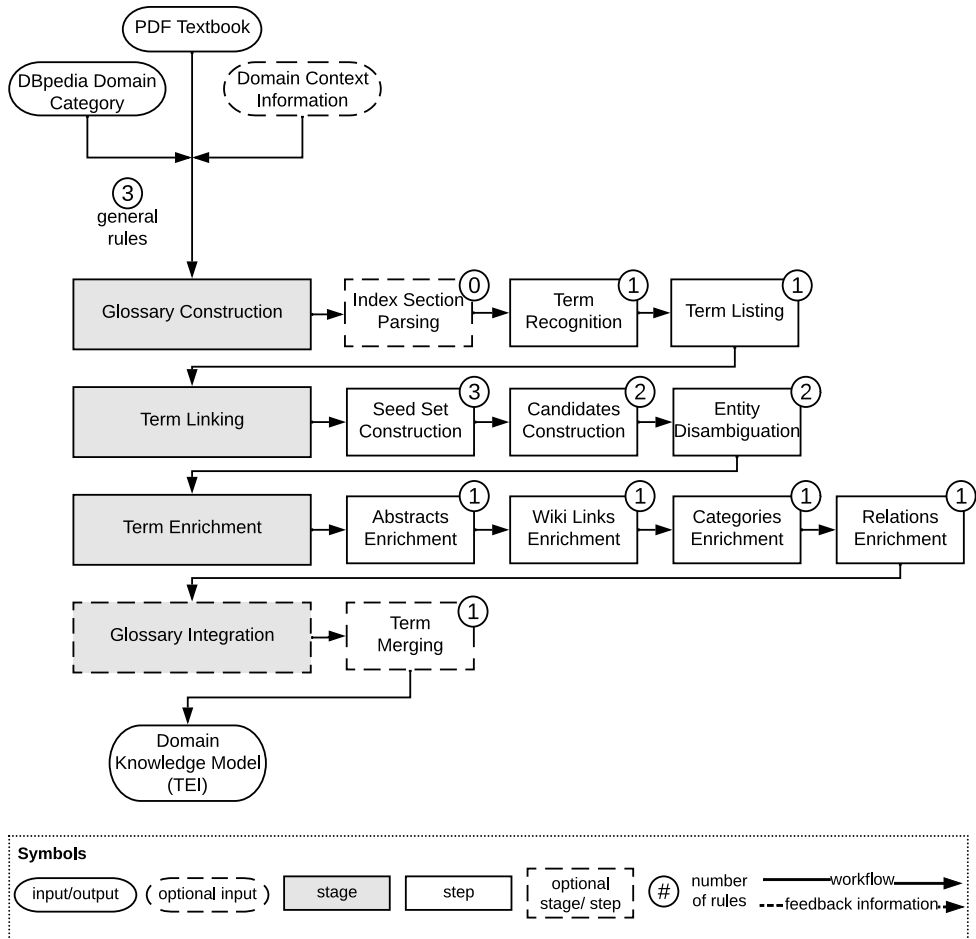


Figure 4.1: Outline of the implemented approach (linking/enrichment and integration phases).

Figure 4.1 shows the overall workflow of the approach, including its four main stages, ten steps, and 17 rules. As the first stage, the approach formally represents all the index terms as a machine-readable glossary linked to pages across the textbook. The next stage extracts DBpedia entities that belong to the target domain of the textbook and links them with relevant concepts from the extracted glossary. After that, the glossary is enriched with additional semantic information from DBpedia, which creates a bridge between the textbook and the Linked Open Data cloud. The only manual intervention that the approach requires is the selection of a DBpedia category matching the domain of the textbook. The approach's last stage is to add more textbooks from the same domain. Integrating multiple textbooks should result

not only in a joint hyperspace of cross-linked textbooks, but also in a more complete and objective model.

4.3.1 General Rules

The workflow has three general rules. `DBPEDIA_ENTITY` is used to query a term in DBpedia and get its URI⁸ if it exists. The rule applies the convention for entities in DBpedia (first capital letter and words separated by an underscore), and it also tries different variations of the term: singular form, lower and upper cases, without accents, without punctuation, and changing any quotes to apostrophe. If any of the term variations corresponds to a subject entity in DBpedia, then a matching entity has been identified.

`FINAL_ENTITY` says that if an entity has the DBpedia redirect property (*dbo:wikiPageRedirects*), the object value of the property is the final URI of the entity. Redirects are used in DBpedia to indicate the final URI of an entity and include synonyms, common misspellings, and acronyms. For example, the entity *dbp:Gaussian_distribution* is redirected to *dbp:Normal_distribution*.

The last general rule, `AMBIGUOUS_ENTITIES`, is used to create a set of entities (*AMB*) related to an ambiguous entity. An entity is ambiguous if one of two possible cases happens. Case ONE is when the found entity has the *dbo:wikiPageDisambiguates* property. For example, the *dbp:Distribution* entity links to 33 entities using the *dbo:wikiPageDisambiguates* property. Case TWO is when the entity does not have the *dbo:wikiPageDisambiguates* property, but is listed as an object with others in an entity that uses that property. For example, the *dbp:Median_lethal_dose* entity appears in the *dbp:MLD* entity through the *dbo:wikiPageDisambiguates* property. If either case applies, the rule generates a set with all the listed entities using the disambiguation property.

Two functions are used by several rules. *q*(·) receives a subject, a property, and an object value to execute a select SPARQL query against DBpedia. One of the three parameters is a variable (marked as ?), which indicates to the function which values should be returned. The function *r*(·) returns the URI of an entity associated with a glossary term. The returned URI corresponds to the final entity after any redirect has been followed. In the case that a term has multiple candidate entities, the function returns the URI of the chosen entity after the disambiguation step.

4.3.2 Glossary Construction

An index section provides a natural source for harvesting important terms in a domain of a textbook. The two main challenges one needs to address when creating a glossary from an index section of a textbook are index extraction and term recognition. The former refers to the task of parsing a textbook and extraction of its index terms, and the latter refers to the identification of the right reading order for each term.

⁸For example, the entity about the *arithmetic mean* has the full URI http://dbpedia.org/resource/Arithmetic_mean corresponding to the shorter (namespaced) version *dbp:Arithmetic_mean*. For simplicity, the rest of this chapter (and the thesis) uses namespaced URIs of entities and properties. Namespace prefixes can be found at <https://dbpedia.org/sparql?help=nsdecl>

Index Section Parsing

Regardless of the format of the textbook (e.g., PDF or EPUB), the first step for glossary construction is to parse the textbook, recognize the index section, and retrieve the index terms. Depending on the textbook format and the index section layout (multi-column, hierarchically structured, or multi-line entries), this process can be less or more challenging. I represents the final list of index entries. This task is a prerequisite for the proposed approach, which is out of the scope of this chapter. In the rest of this thesis, angle brackets ($\langle \rangle$) are used to separate the hierarchical parts of the index terms (written in *italics*).

Term Recognition

In this step, the reading label for each index term is identified to be used in the glossary. Since grouped index terms do not have a standardized reading order, the reading labels identify the form in which the index terms appear in the textbook's content. The rule `READING_LABEL` identifies the right reading label (r_i) of each index term ($i \in I$). For example, the hierarchical index entry *distribution* $\langle \rangle$ (*gamma, normal*) has three index terms: *distribution*, *distribution gamma* (with possible labels *distribution gamma* and *gamma distribution*) and *distribution normal* (with possible labels *distribution normal* and *normal distribution*). For those terms, *gamma distribution* and *normal distribution* are the proper reading labels since those are the forms used in the text. This step is not straightforward because sometimes the index terms appear on the pages written in a different form (e.g., singular vs. plural form, or with a different conjugation for verbs). Simple textual search is not enough to detect the different variants of index terms in a book. For example, a textual search will fail to find the *Bernoulli distribution* index term in the sentence "Bernoulli and binomial distributions", but the rule is able to detect such an occurrence. `READING_LABEL` is a term recognition algorithm (*TRA*) that checks the possible labels ($LABELS_i$) of an index term (i) against its reference sentences ($SENTENCES_i$) (according to the locators of the index entry) to identify the right labels. $LABELS_i$ are created permuting all the parts or lines of a hierarchical index entry. Then, each label (lab) and each sentence (sen) from the text are broken into noun chunks. A noun chunk is a simple phrase with a noun as its head. Each noun chunk has a noun plus the words describing the noun (e.g., "autonomous cars"). Breaking the text into noun chunks helps discovering meaningful words that are equal in both the candidate labels and the sentences, and discarding auxiliary phrase parts like articles. Words in a noun chunk that are not adjectives, verbs, nouns, or proper nouns are discarded. After the candidates and the sentences are broken into noun chunks, the rule tries to identify the labels in each of the sentences by comparing the noun chunks. The function $c(\cdot)$ is used to compare two noun chunks (ch and ch') employing their lemma and stem forms. Lemmas are the canonical or dictionary forms of words, and word stems are the root forms of words. Their use is the key factor that allows the algorithm to find the right labels even if a term is written differently. In $c(\cdot)$, the lemma form of words are compared, and if there is no match, the word stems are compared. If all the tokens from the noun chunks of the candidate appear in the same order in one sentence, the possible label ($lab \in LABELS_i$) is marked as the right reading label (r_i) for the index term:

$$\begin{aligned} &\text{if } \exists lab \in LABELS_i \text{ and } \exists sen \in SENTENCES_i \\ &\text{and } \forall ch \in lab \exists ch' \in sen : c(ch, ch'), r_i \mapsto lab. \end{aligned} \quad (4.1)$$

Term Listing

In this step, the index entries and the recognized reading labels are used to construct a glossary of terms G . For each index term, a glossary term g_i is added to G . Each glossary term corresponds to a list $(property_1, \dots, property_n)$ formed by properties. A property is new information associated with an index term. Initially, a glossary term contains the main label ($main_i$) and a set of alternative labels (ALT_i). The rule TERM_LABELS is used to list for each term $i \in I$, its different labels, formally:

$$\begin{aligned} &\text{if } \exists r_i, main_i \mapsto r_i \text{ and } ALT_i \mapsto \emptyset; \\ &\text{otherwise, } main_i \mapsto natural(i) \text{ and } ALT_i \rightarrow permu(i) - natural(i). \end{aligned} \quad (4.2)$$

The function $natural(\cdot)$ returns the name as it appears in the index section (read from top to bottom and from left to right), and the function $permu(\cdot)$ calculates all possible permutations of the hierarchical parts of an index term. For example, if $i_0 = \text{adjacent value } \langle \rangle \text{ lower}$, $natural(i_0) = \text{adjacent value lower}$ and $permu(i_0) = \{ \text{adjacent value lower, lower adjacent value} \}$. At this moment, $g_i = g_i \cup main_i \cup ALT_i$.

Sometimes authors index the same term in several ways. For example, the term *normal distribution* sometimes appears as two index entries: *normal distribution* and *distribution* $\langle \rangle$ *normal*. In those cases when the right reading order of both terms is the same (e.g., *normal distribution*), they appear as one unique term in G .

At the end of this step, an enrichment model E of the textbook T is created. Initially, $E = (G)$.

4.3.3 Term Linking

In this stage, the terms from the constructed glossary G are linked to DBpedia entities. Finding the right entity for a term involves querying DBpedia and constructing a list of possible candidates, and then using a disambiguation strategy to choose the best match. Initially, a seed set of terms that only have one possible matching entity in DBpedia is constructed from the glossary. Then, the terms that have multiple candidate entities are identified and marked to be disambiguated. Finally, the disambiguation strategy computes cosine similarity between the extended abstracts of candidate DBpedia entities and the provided context text to find the corresponding entity from the list of candidates. If no initial context information is provided, the abstracts in DBpedia from the entities in the seed set are used as context. This stage has three steps.

Seed Set Construction

In this step, a seed set of terms $SEED$ is created. This set includes the terms from G for which only a single DBpedia entity has been found, which is unambiguously the right entity for the term. The entities are found with the help of the DBpedia

categories. Categories⁹ correspond to a special type of entities in DBpedia used to classify and group together entities on similar subjects. They are ordered in a broad and non-strict hierarchy (i.e., sub-categories can have multiple super-categories). The idea is that if a term has only one possible candidate entity in DBpedia and that entity belongs to a (sub-)category that is a part of the target DBpedia domain; it is safe to link the term to such entity unambiguously.

Inspired by Mirizzi et al. [198] and Slabbekoorn et al. [249], the main DBpedia domain category given as input (cat_{main}) is used to find a set of (sub-)categories that belong to the domain of the glossary, defined as CAT . Using the *skos:broader* property, it is possible to query DBpedia and get all categories corresponding to a domain. For example, 13 subcategories one level down in the hierarchy are obtained when querying the *dbc:Statistics* category. They include *dbc:Statistical_models*, *dbc:Applied_statistics*, *dbc:Statistical_theory*, and *dbc:Sampling_(statistics)*. This process can be done recursively to extract categories from increasingly deeper sub-levels of the hierarchy. As the number of categories increases exponentially, the chance to select marginally relevant or even irrelevant entities for the seed set increases fast. After running several experiments, the number of recursive levels has been set to three. Initially, $CAT = \{cat_{main}\}$. Then, the set is updated according to the rule EXPANDED_CATEGORIES:

$$CAT = CAT \cup \{q(?, skos:broader, cat) : cat \in CAT\}. \quad (4.3)$$

After the set of in-domain categories has been constructed, each $g \in G$ is checked to see if it belongs to the seed set using the three general rules and an additional rule. First, DBPEDIA_ENTITY is used to query DBpedia and get the corresponding entity of g . Then, if an entity was found, the rule FINAL_ENTITY is used to get the final URI of the entity. After that, the rule AMBIGUOUS_ENTITIES is used to check if the entity is not ambiguous, meaning that the $AMB_{r(g)}$ set is empty. Finally, the rule SEED_SET_ENTITY uses the *dct:subject* property to get the categories of the entity and decide if the term belongs to $SEED$:

$$\text{if } |AMB_{r(g)}| = 0 \text{ and } \exists cat \in q(r(g), dct:subject, ?) : cat \in CAT, g \in SEED; \quad (4.4) \\ \text{otherwise, } g \notin SEED.$$

At the end of this step, if no domain context information was supplied as input, it is constructed. The rule DOMAIN_CONTEXT_INFORMATION says that the set with all abstracts from each term in $SEED$ forms the domain context information $DOMAIN$. Abstracts are retrieved using the *dbo:abstract* property and the $q(\cdot)$ function.

Now, $E = (G, CAT, SEED, DOMAIN)$.

Candidates Construction

After the seed set has been created, the remaining terms in the glossary still need to be linked to entities in DBpedia. In this step, a list of candidate entities to be associated to each of those remaining terms is constructed. *can*d is defined as a candidate

⁹https://en.wikipedia.org/wiki/Help:Category_services-resources/datasets/dbpedia-datasets#h434-7

<https://wiki.dbpedia.org/>

entity that is associate to a $g_i \notin SEED$. $CAND_x = \{cand_{1x}, cand_{2x}, \dots, cand_{nx}\}$ corresponds to a list of candidates $list_x$ formed by candidate entities. The list $LIST = (list_1, list_2, \dots, list_n)$ corresponds to all the candidate lists from all the glossary terms that do not belong to $SEED$. Each candidate $cand$ has the URI of the entity uri and its context information ctx .

For each $g_i \notin SEED$, the three general rules are used to find a matching DBpedia entity and get the set of related entities AMB if the term is ambiguous. One example of an ambiguous term is *mean*. DBpedia returns the *dbr:Mean* entity when querying the term. This entity does not have the *dbo:wikiPageDisambiguates* property, but it appears in a group with other entities in the *dbr:Mean(disambiguation)* entity using the *dbo:wikiPageDisambiguates* property¹⁰. Figure 4.2 shows the list of related entities for the term *mean* after resolving the disambiguation property using the rule `AMBIGUOUS_ENTITIES`.

- 
- *dbr:Mean*
 - *dbr:Ethic_mean*
 - *dbr:MEAN_(software_bundle)*
 - *dbr:Mean_(album)*
 - *dbr:Meane*
 - *dbr:Meanness*
 - *dbr:Means_(disambiguation)*
 - *dbr:Mean_(song)*
 - *dbr:Mean_(magazine)*

Figure 4.2: Candidate list for the term *mean*.

Then, the list of candidates is created using `CANDIDATE_LIST`. This rule says that the matching DBpedia entity and all the entities from the set of related entities are the candidates of the term, which are added to $CAND_x$. When each candidate entity *cand* is created, its context information is retrieved to be used in the disambiguation process. Since the disambiguation stage uses cosine similarity, the primary context information for an entity is its abstract. Additionally, based on other approaches [192, 208] that gather context information such as all paragraphs mentioning a candidate entity in Wikipedia or the titles of other entities that link to the candidate, `EXTENDED_ABSTRACT` creates a piece of context information for the candidate. The rule uses the *dbo:wikiPageWikiLink* property to find all other entities where their Wikipedia page links to the page of the candidate (*cand*) and then, it creates the context information of the candidate (*ctx*) by concatenating all the abstracts:

¹⁰[http://dbpedia.org/resource/Mean_\(disambiguation\)](http://dbpedia.org/resource/Mean_(disambiguation))

$$\begin{aligned}
 LINKS &= q(?, \text{dbo:wikiPageWikiLink}, cand); \\
 ctx &= q(cand, \text{dbo:abstract}, ?) \cup \{q(link, \text{dbo:abstract}, ?) : link \in LINKS\}.
 \end{aligned}
 \tag{4.5}$$

The assumption is that the candidate entities that belong to the same domain as the term g_i will get abstracts also from other entities in the domain, and this will boost the candidate when applying cosine similarity in the next stage.

The list of all the candidate lists $LIST$ is added to E . Now, $E = (G, CAT, SEED, DOMAIN, LIST)$.

Entity Disambiguation

The final step of the current stage is to apply a disambiguation strategy to choose the best entity in each $CAND_x$. Two rules use the glossary G , the list of all candidate lists $LIST$, the DBpedia domain category cat_{main} , the domain context information $DOMAIN$, and a threshold th in this step. The strategy is to compare the context information of each candidate with the domain context information using a similarity function based on cosine similarity and to select for the term the entity with the highest similarity score that surpasses the minimum threshold. The selection of the entities is incrementally to first select the entities that are more likely to be the right match.

`SIMILARITY_SCORE` computes a similarity score sim for each candidate entity $cand$ in a candidate list $CAND_x$. First, the standard cosine similarity between ctx and $DOMAIN$ is calculated. Sometimes candidates for a term are closely related, and depending on the extended context information for each of them, the wrong one can be chosen. Due to this scenario, the rule has three cases where the cosine similarity value is modified to produce sim . Case ONE assigns $sim = 1$ to a $cand$ if it has the same name as its term g_i , the domain of $cand$ is cat_{main} , and its cosine similarity value is higher than th . The function $d(\cdot)$ uses the fact that some entities have explicitly encoded the domain for which they belong in the URI to detect the main domain of $cand$. For example, when querying the candidates for the term *Method of moments* for a glossary in the *statistics* domain, two candidate entities are retrieved: `dbr:Method_of_moments_(statistics)` and `dbr:Method_of_moments_(probability_theory)`. In this example, the `dbr:Method_of_moments_(statistics)` entity gets a similarity value of 1. Case TWO assigns $sim = 0.9 + (sim/10)$ to a $cand$ if it only has the same name as its term g_i (there are no other candidates with the same name), $cand$ does not have an explicit domain in its URI, and its cosine similarity value is higher than th . Case THREE assigns $sim = 1.2 * sim$ to a $cand$ if it is the entity returned when querying DBpedia with the rule `DBPEDIA_ENTITY`, and it is not a disambiguation page. In the *mean* example, from the candidate list of entities (Figure 4.2) the similarity score of the `dbr:Mean` entity gets an increase of 20%. Case TWO and THREE seek to increment the chances of the candidate to be chosen, taking into account the obtained cosine similarity value.

`SELECTED_ENTITIES` chooses the right entities from the candidates in an incremental process. First, the rules use `SIMILARITY_SCORE` to calculate the sim value

for each *cand*. Then, the *cand* with $sim = 1$ in each $CAND_x$ is selected. After that, the abstracts of the chosen entities are added to *DOMAIN*. Then, the cycle starts again, but now the minimum *sim* for a candidate to be selected is 0.9. The process continues decreasing the minimum similarity score by 0.1 each time until the given threshold *th* is reached. The idea behind this rule is that by first selecting the entities that get a higher similarity score, *DOMAIN* is expanded with the context information from the selected entities. This constant expansion of *DOMAIN* will help in the next cycles of the rule to distinguish the candidates that belong to the same domain from the rest and select the right entity for each term.

At the end of this step, the terms from the glossary with an identified matching DBpedia entity are saved as the list of selected entities *SLCT*. Now, $E = (G, CAT, SEED, DOMAIN, LIST, SLCT)$.

4.3.4 Term Enrichment

The third stage is to use the discovered DBpedia entities to extract other semantic information and enrich the terms from the glossary. The enriched glossary can be seen as a rich domain model that contains representative concepts in a domain. The rules in each of the steps in this stage enrich each $g_i \in (SEED \cup SLCT)$.

Abstracts Enrichment

The rule *ABSTRACT* adds the abstract from its matching DBpedia entity to each term: $g_i = g_i \cup q(r(g_i), \text{dbo:abstract}, ?)$. Abstracts bring summarized definitions for the index terms of the textbooks.

Wikipedia Links Enrichment

The Wikipedia pages from where the information was extracted to create the entities in DBpedia are used to enrich the terms further. The rule *WIKIPEDIA_LINK* uses the *foaf:primaryTopic* property of the entities to enrich each term with the link from its Wikipedia page: $g_i = g_i \cup q(r(g_i), \text{dbo:foaf:primaryTopic}, ?)$. This enrichment allows the model to quickly redirect a user who is navigating the textbook model to the corresponding Wikipedia page of the concepts (index terms).

Categories Enrichment

The *dct:subject* property is used by the rule *CATEGORIES* to enrich each term with its DBpedia categories: $g_i = g_i \cup q(r(g_i), \text{dct:subject}, ?)$. Adding categories to the terms will allow the model to have more information to create relations between the terms of the glossary. For example, the categories can be used to create clusters of terms that are related or to create a map of the different sub-domains that are part of the textbook.

Relations Enrichment

Finally, the information about linked Wikipedia pages stored in DBpedia are exploited to discover relations among the terms in the glossary. The general idea is that if two entities in DBpedia are linked together, the corresponding terms in the glossary are also related. For each term with a linked entity, the rule *RELATIONS* queries DBpedia uses the *dbo:wikiPageWikiLink* property to retrieve all other entities that the current entity links to in its corresponding Wikipedia page. Then, for

each retrieved entity that is linked to a term, a relation is created between the two terms. Formally:

$$\begin{aligned} LINKS &= q(r(g_i), \text{dbo:wikiPageWikiLink}, ?); \\ REL_x &= \{g_y : g_y \in G, r(g_y) \in LINKS\}; \\ g_i &= g_i \cup REL_x. \end{aligned} \tag{4.6}$$

For example, the term *mean* is linked to the entity *dbp:Mean*, which links to 87 entities. One of those entities is *dbp:Mode_(statistics)*. If in the glossary the term *mode* is linked to *dbp:Mode_(statistics)*, then a relation between *mean* and *mode* is created.

For the moment, two terms are only related in a general manner without detailing the type of relationship. In the future, the idea is to exploit both the hierarchical structure of index terms and the content in textbooks to be able to discover subtopic hierarchies [27] and more specific relations (e.g., *is-a* relations) [165].

4.3.5 Glossary Integration

The final stage is optional and is used to integrate the terms from multiple textbooks in one glossary. This stage has only one step.

Term Merging

First, for each textbook term, a special entry is created in a combined glossary. Each entry has a Preferred Label (PF), a set of Alternative Labels (AL), and an External Concept (EC). The first one corresponds to the ID of the term, the second to a set of alternative names for the same term, and the last to an external concept from a different glossary or ontology. For example, a glossary term is {PF: *normal distribution*; AL: *distribution* () *normal*, *Gaussian distribution*; EC:- }. Initially, no term has an external concept; this association is done in specific cases.

Since two terms representing the same concept can be written differently, all entries are analyzed to identify repeated terms. Terms that are identified as the same are combined, keeping the individual terms as alternative labels. The rule SAME_TERMS identifies semantically equal terms using four cases: (1) exact textual matching; (2) different lexical forms (e.g., *distribution* and *distributions*); (3) acronyms (e.g., *ANOVA* and *Analysis of variance (ANOVA)*); and (4) synonyms (e.g., *student's test* and *t-test*). The first case is trivial. The second case is handled by comparing the terms using their stems (computed for each word using the snowball stemming algorithm [222]). In the third case, acronyms are handled by first identifying if an index term contains some text between parenthesis and then splitting the term into two. Each part of the term is compared against the other terms. Finally, synonyms are handled with the help of DBpedia: terms linked to the same DBpedia entity during the enrichment process are merged. Additionally, terms linked to the same external concept (if any) are also merged. The *dbo:wikiPageRedirects* property of the DBpedia is also used, which indicates synonyms, common misspellings, and acronyms to increase the matching between the terms. The result of this step is one unified glossary with the index terms from all the textbooks.

4.3.6 Enriched TEI model

The previous chapter (🔗 Chapter 3) has described how to construct a knowledge model of a textbook using the TEI. Using the information from *E*, it is possible to enrich the original textbook model to incorporate additional semantic information and create connections to the LOD using RDF relationships. Since the TEI Guidelines do not provide a mechanism to integrate RDF with TEI¹¹, RDFa¹² attributes are added to the TEI attribute class `att.global linking: @about, @property and @resource` [235, 264]. Those attributes allow for representing RDF tuples: `@about` for the subject, `@property` predicate and `@resource` object. Table 4.1 presents the additional mapping between the extracted semantic information and the used TEI elements.

4.4 Evaluation

Three evaluations of the approach have been conducted. In the first and second evaluation, the quality of the first two stages of the approach was tested to see whether the terms are linked to the correct entities in DBpedia using a manually constructed ground truth. The last evaluation examined how adding more textbooks would affect the ability of the approach to connect DBpedia entities from a target domain. All the evaluations were conducted using a local copy of DBpedia (version 2016-10¹³).

4.4.1 Precision of Term Recognition

The goal for this evaluation was to find out how many of the index terms extracted from the textbooks are recognized in the pages of the textbooks using the described term recognition algorithm. In this evaluation, the number of recognized index terms and reference pages from the algorithm were compared against a baseline. The hypothesis is that using Part-of-speech tagging to recognize the nouns in the index terms would increase the number of recognized index terms in the textbook pages compared to textual search.

Procedure

For this evaluation three textbooks have been used. The first two in the statistics domain: *STAT#1* [69], and *STAT#2* [282]. The third book is for information retrieval: *IR#1* [177].

First, for each of the textbooks, the index terms and the reference pages (locators) have been extracted from their knowledge models. Index terms are the names of each index entry, and the reference pages are the pages in the textbooks annotated with an index term; these reference pages appear in the form of page numbers next to the index terms in the index section. Then, the term recognition algorithm (*TRA*) has been applied to the index terms and reference pages, as used by the

¹¹There is an open discussion around the topic among the TEI community, see <https://github.com/TEIC/TEI/issues/1860> [accessed 03-2020].

¹²<http://rdfa.info/>

¹³<https://wiki.dbpedia.org/downloads-2016-10>

Table 4.1: Map between the semantic information and the TEI textbook model.

Category	Textbook component	TEI elements	TEI attributes / values	Notes
Semantic information $(g_i \in G)$		$\langle \text{seg} \rangle$ (arbitrary segment) + $\langle \text{gloss} \rangle$ (gloss or definition) + $\langle \text{label} \rangle$ + $\langle \text{ref} \rangle$ (reference)	$\text{@about} = \text{"[URI]"} +$ $\text{@property} = \text{"[namespace:property]"} +$ $\text{@resource} = \text{"[URI]"} +$	$\langle \text{seg} \rangle$ groups the info for a term: $\langle \text{gloss} \rangle$ is used for abstracts, $\langle \text{label} \rangle$ for reading label, $\langle \text{ref} \rangle$ for Wikipedia pages, categories, and term relations

READING_LABEL rule. A baseline strategy was used for comparison. The *baseline* (*BL*) tried all possible labels of each index term (☞ Section 4.3.2) using simple textual search to find the index term in the reference pages. The baseline assumes that the form of the index term is the same as the form used in the actual content of the chapters, which is not always the case. Table 4.2 shows the results. The following values are reported: the percentage of index terms where the label (the right reading order) is recognized in at least one reference page (% index terms) and the percentage of reference pages where the index term is recognized (% reference pages).

Analysis

Table 4.2: Term recognition.

Metric / Textbook		STA#1	STA#2	IR#1
# Index Terms		651	591	608
# Reference Pages		738	859	774
BL	% Index Terms	63.44	82.74	87.66
	% Reference Pages	62.60	83.70	85.27
TRA	% Index Terms	82.64	93.06	94.08
	% Reference Pages	81.44	92.89	92.64

Results show that the performance of the term recognition algorithm is more effective than the baseline in both index term and reference page recognition. In the STAT#1 textbook, the term recognition algorithm increases the number of recognized index terms by 30.26 % and by 30.10 % the number of pages where the index terms are recognized, taking the *BL* as the reference point. The algorithm achieves the smallest increases in the IR#1 textbook, by 7.32 % and by 8.64 % respectively.

The algorithm can effectively detect index terms that appear fragmented in the text pages (☞ Section 4.3.2, the *Bernoulli* example), but its performance decreases recognizing terms written using synonyms in the pages. For example, in STAT#1, the term *Agresti-Coull method* appears as *agresti-coull interval* in its reference page. Another case where the algorithm cannot recognize the index terms is when not all words of the index terms are used. For example, the term *XML Tag* in IR#1 appears only as *tag* in its reference page. Also, it happens that an index term does not appear in one of its reference pages; the reference is only used to indicate that the topic of that page is related to the index term. It is planned to add an external model (e.g., a dictionary) to deal with synonyms, and increase the flexibility of the algorithm to recognize different parts of the index terms.

4.4.2 Precision of Linking

In this second evaluation, the goal has been to find out if the index terms with candidate entities in DBpedia are linked to the right entities. This evaluation compares the number of correctly selected entities using the proposed approach against two baselines. The belief is that the construction of a seed set of entities, and the use

of the context information extracted from the entities in that set, as in the proposed approach, will perform better at disambiguating and choosing the right entity for each term than approaches that do not use context information.

Procedure

To validate the hypothesis, the linking of the index terms to DBpedia has been tested using the same three books from the previous evaluation (STAT#1, STAT#2, and IR#1). For STAT#1 and STAT#2, the approach was used with the *dbc:Statistics* category given as input to indicate the domain of the textbooks. For IR#1, the given category was *dbc:Information_retrieval*. In none of the three cases was domain context information given; instead, it was extracted using the created seed set. After the "Seed Set Construction" and "Candidates Construction" steps, there were 47 terms in the seed set for STAT#1, 67 for STAT#2, and 43 for IR#1. The number of terms with candidates lists were 146 for STAT#1, 164 for STAT#2, and 306 for IR#1.

For the entity disambiguation step three methods were used: the proposed strategy and two baselines that do not use context information for disambiguation. Based on Mendes et al. [192], the baselines are:

- *Random Baseline (BL1)* selected randomly one of the entities in the candidates list as the right entity. This baseline will show if the proposed algorithm selects the right entity better than chance.
- *Default Sense Baseline (BL2)* selected the entity in the candidates list which is the most referenced entity in Wikipedia from other pages. This baseline will show the impact of using a specific domain for disambiguation in contrast to choosing just the most used entity in the candidates list.

For evaluating the selected entities, a ground truth was created manually by marking from each of the candidates lists the right entity. Precision is defined as the number of **correctly selected entities** from the candidates lists divided by the number of **selected entities** from the candidates lists. Recall was computed dividing the number of **correctly selected entities** by the number of **relevant entities**. The total number of relevant items corresponds to the number of terms that belongs to the domain of the textbook and has a matching entity in their candidates lists. Since the proposed strategy uses a minimum threshold for selecting a candidate as the right entity, the evaluation also explored which value would give the best results. Figures 4.3, 4.4, and 4.5 show for each textbook the precision and recall of the proposed strategy using different thresholds, and the precision and recall using the two baselines. The values for BL1 are the average of 100 repetitions. The values for the baselines are constant since they are not affected by a threshold.

Analysis

As expected, the performance of the random baseline is the lowest of the three strategies, which confirms that a disambiguation algorithm is needed since chance does not yield good results. The default sense baseline performs better for the three books, reaching up to 0.47 and 0.56 for the precision and recall of IR#1, respectively. Nevertheless, the proposed strategy obtains the best results. The use of context information achieves better disambiguation than just selecting the most used entities

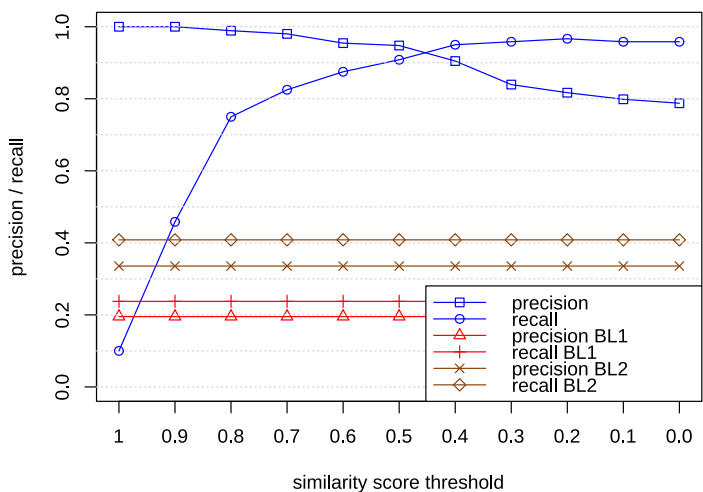


Figure 4.3: Precision and recall for the linking evaluation of STAT#1 textbook.

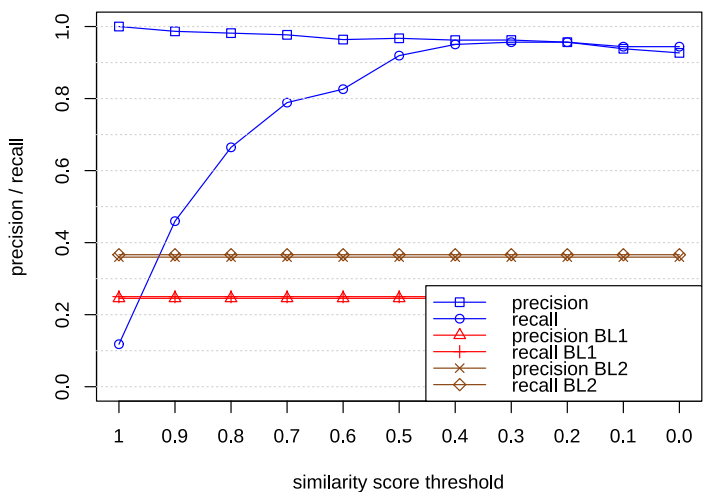


Figure 4.4: Precision and recall for the linking evaluation of STAT#2 textbook.

because the terms of the glossary belong to the same domain, and the proposed strategy exploits this characteristic by constructing the seed set of entities. For the STA#1 precision and recall are balanced when the minimum threshold is set between 0.4 and 0.5. For STAT#2, the balance is achieved with a threshold up to 0.5. For the information retrieval book, the values between 0.5 and 0.6 for the threshold get the best balance for precision and recall. By manipulating the threshold value, it is possible to favor precision or recall depending on the final use of the algorithm. In most use cases, higher precision (given a reasonable recall) is preferred to minimize

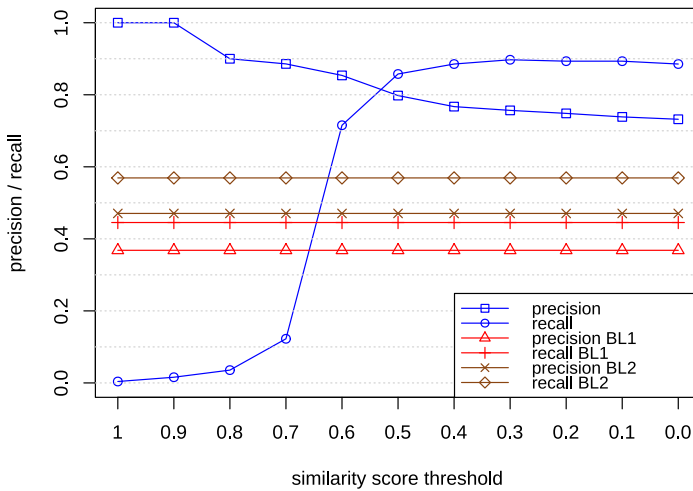


Figure 4.5: Precision and recall for the linking evaluation of IR#1 textbook.

the number of errors in the model. In general, as the threshold decreases and gets closer to 0, the algorithm selects not only incorrect entities, but also irrelevant ones; they do not belong to any of the related domains of the textbooks.

Additionally, two other evaluations were carried out given specific context information and not using the information constructed from the seed set. The context information was obtained using the textbooks. First, for each index term, the sentences in which the term appears were extracted. All the sentences concatenated were given as the context information. Also, other context information was created by extracting the paragraph, instead of just the sentence, where the index terms appear in the textbooks. The results obtained using the context information from the seed set and the context information from the sentences and paragraphs of the textbooks are comparable. The mean absolute error for the precision fluctuates from 0.015 to 0.033 depending on the threshold and the context, and for the recall the mean absolute error fluctuates from 0.022 to 0.156. These evaluations show that the context information extracted from the seed set is as useful for the disambiguation algorithm as the context directly taken from the textbooks.

4.4.3 Discovery of entities

The goal of the final evaluation experiment was to determine the effect of adding more textbooks in the glossary construction task and test how many entities the approach will find in DBpedia for a target domain. This evaluation used a ground truth of DBpedia entities in the statistics domain to quantify the entities in the domain that can be discovered. The belief is that by integrating glossaries from multiple textbooks in the same domain it is possible to build a consolidated model providing a more complete and objective representation of the domain knowledge. In more practical terms, with every new textbook integrated, the consolidated model should

better approximate the ideal model, which is in this case, the subset of DBpedia entities that belong to the same domain as the textbooks.

Procedure

To test the hypothesis, ten introductory statistics textbooks have been used [64, 69, 72, 90, 101, 144, 174, 246, 272, 282]. The idea was to find out how many of the linked entities belong to the statistics domain, and compared that to the total number of entities that belong to the domain. For this evaluation, precision is defined as the number of **linked entities that belong to the domain** divided by the number of **linked entities**. The recall was the number of **unique linked entities that belong to the domain** divided by the **total number of entities in DBpedia that belong to the domain**. To determine the actual entities in DBpedia that belong to the statistics domain a ground truth using the ISI Glossary. The proposed approach was used to link the terms from the comprehensive ISI Glossary to DBpedia entities, and after manually checking the obtained linked terms, a ground truth of 1049 entities in the statistics domain was obtained.

Three configurations were evaluated using the ten textbooks. Figure 4.6 presents the results for entity discovery using individual textbooks; the results are averaged across all ten textbooks to account for potential differences between them. Figure 4.7 shows the average discovery of entities of two sets of five textbooks each. Finally, Figure 4.8 shows the discovery of entities when using all of the ten textbooks combined. For each configuration, a unified glossary was created (☞ Section 4.3.5).

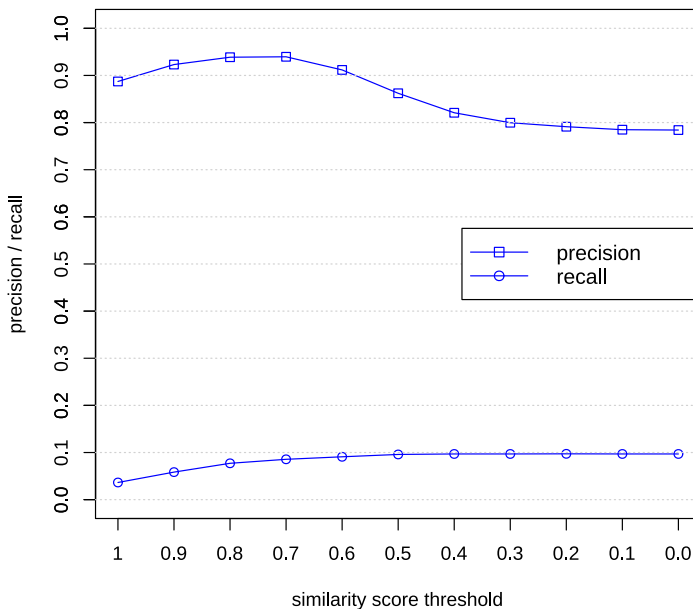


Figure 4.6: Precision and recall for entity discovery using one textbook (averaged over ten books).

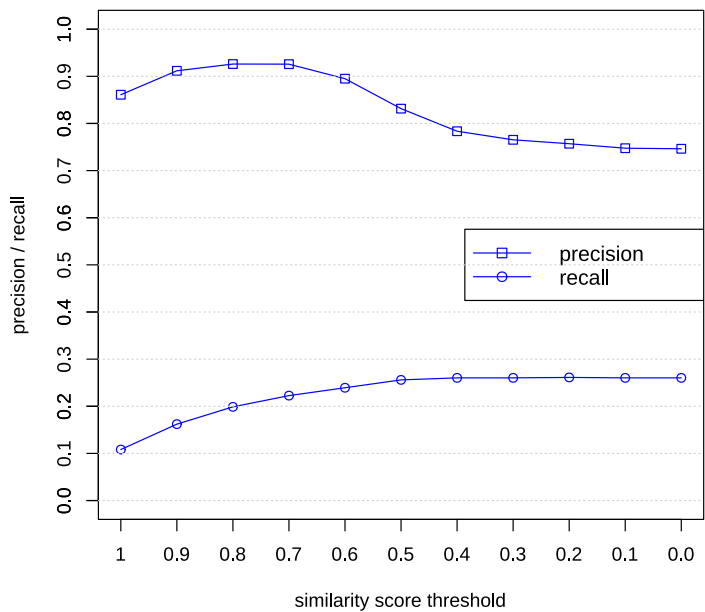


Figure 4.7: Precision and recall for entity discovery using five textbooks (averaged over two sets of five books).

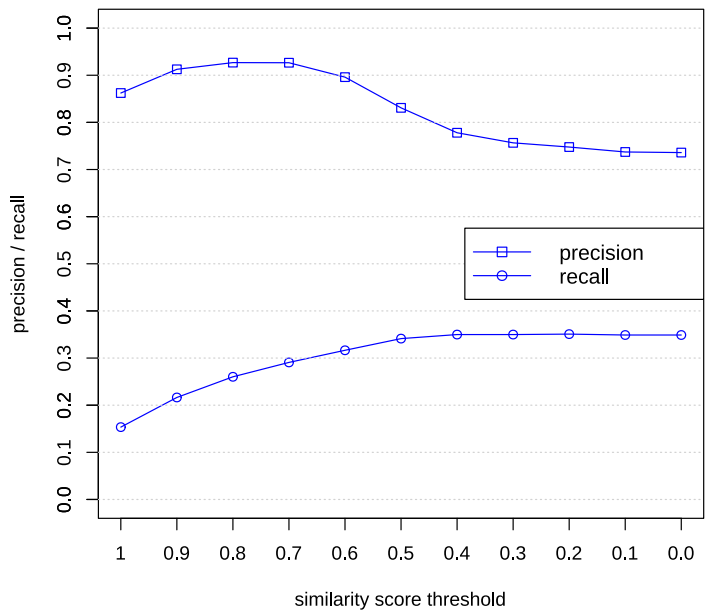


Figure 4.8: Precision and recall for entity discovery using ten textbooks.

Analysis

When using just one textbook, the average precision varies from a maximum of 0.940 to a minimum of 0.784, while the average recall grows from 0.036 to 0.097 (Figure 4.6). The recall is rather low, as a single textbook provides a limited number of glossary terms (average of 512) of which only a portion has candidate entities in DBpedia (average of 132). The second configuration of five textbooks in two sets has an average of 2353 glossary terms and an average of 511 terms with candidates in DBpedia. As shown in Figure 4.7, as the number of terms increases, the recall goes up, reaching up to 0.261, but the precision goes down to the minimum value of 0.746. The final configuration (Figure 4.8) which includes ten textbooks and 4455 terms (854 with candidate entities) gets a recall of 0.350, which means that more than a third of all entities are found using only textbooks in one part of the domain: introductory statistics.

The reached recall value can appear low, but it is directly related to the broad coverage of the ISI glossary. In order to discover more entities in the domain, additional textbooks from a more diverse and applied side of statistics should be used. For example, the term *cox's theorem*¹⁴ is a statistical theorem, it is not part of any of the ten textbooks, but it belongs to the target domain. Using Google Books, this term was found in three textbooks with particular topics: uncertainty theory¹⁵, statistical evidence measurement¹⁶, and universal artificial intelligence¹⁷. All three books mention the term, but it appears only in the index section of the last one, which illustrates the rarity of the term.

In all of the three cases, the reduction of the precision as the recall increases is explained by the fact that the index sections of the textbooks contain terms that are not only related to statistics but also to other domains like probability and general mathematics. For example, the entities *dbr:Slope*, *dbr:Law_(stochastic_processes)*, and *dbr:Logarithmic_scale* are linked to terms in the configurations, but those entities do not correspond to terms from the ISI glossary; therefore, they are not part of the ground truth. Filtering linked entities that do not belong to the domain could be accomplished by further exploiting the categories of the entities or by analyzing the graph structure of DBpedia to detect entities that do not belong to the cluster of in-domain entities (🔗 Chapter 5). The obtained recall values support the hypothesis that it is possible to discover all entities in a domain using textbooks as the source of concepts in a domain.

4.5 Conclusions and Future Work

This chapter has looked into textbooks as a ubiquitous, yet underused source of extractable knowledge models. In particular, the focus has been on indices containing manually selected and curated structured glossaries of important terms. When extracted and formally represented these glossaries can become both the backbones of

¹⁴http://dbpedia.org/resource/Cox's_theorem

¹⁵Liu, B. (2007). *Uncertainty Theory*. New York: Springer.

¹⁶Evans, M. (2015). *Measuring statistical evidence using relative belief*. Boca Raton: Taylor & Francis.

¹⁷Hutter, M. (2010). *Universal artificial intelligence sequential decisions based on algorithmic probability*. Berlin: Springer.

knowledge-driven hyper-structures of the textbooks and the semantic bridges allowing to connect textbooks to one another and to the LOD sets and knowledge bases such as DBpedia. Additionally, linking to DBpedia entities allows for connecting the index terms to their corresponding entities (no ambiguity) and obtaining additional semantic information.

Experiments evaluating the proposed approach have demonstrated that even for a single textbook the quality of term recognition and model linking strongly outperforms the baselines. When more textbooks from the same domain are added to generate a composite model, the coverage of model grows significantly. For a threshold value around 0.5-0.6, the recall reaches almost a third of the total, while precision remains as high as 80-90%.

Several directions are planned for future work. The overall approach for textbook model extraction and linking to external models can be further improved. In particular, it is interesting to explore how to maintain high accuracy of the composite model when more and more textbooks are integrated. Some mechanisms to enforce consensus among individual glossaries need to be implemented. Another direction for work is to test this approach across multiple domains. Some differences have been observed when comparing *statistics* and *IR*. It would be interesting to discover the limits of applicability of the approach by trying less formal domains such as history or art. Generated models provide unique opportunities to facilitate information access to the content of textbooks or enrich them with external entities connected to existing open knowledge bases. Exploring practical applications of the approach is an exciting task as well (🔗 Chapter 7). Finally, on a grander scale, this line of research potentially leads towards generation of a global hyperspace of high-quality educational content, where textbooks from the same and relevant domains are linked thematically and the models extracted from these textbooks are built into the global Web of knowledge enabling new generation of information services.

What's in an Index: Extracting Domain-specific Knowledge Graphs from Textbooks

Abstract A typical index at the end of a textbook contains a manually-provided vocabulary of terms related to the content of the textbook. In this chapter, the extraction of knowledge models from digital textbooks is extended. This chapter takes a more critical look at the content of a textbook index. It presents a mechanism for categorizing index terms according to their domain specificity: a *core domain* concept, an *in-domain* concept, a concept from a *related domain*, and a concept from a *foreign domain*. The aggregated linguistic and structural information from textbooks and DBpedia is leveraged to construct and prune the domain-specific knowledge graphs. The evaluation experiments demonstrate (1) the ability of the approach to identify (with high accuracy) different levels of domain specificity for automatically extracted concepts; (2) its cross-domain robustness; and (3) the added value of the domain specificity information. These results clearly indicate the improved quality of the refined knowledge graphs and widen their potential applicability.

This chapter is based on the following publication:

Alpizar-Chacon, I. & Sosnovsky, S., “What’s in an index: extracting domain-specific knowledge graphs from textbooks”, in: *Proceedings of the ACM Web Conference 2022 (WWW ’22)*, 2022, pp. 966–976.

5.1 Introduction

Back-of-the-book indices are collections of terms that can help textbook readers in several ways. As a navigation tool, an index provides readers with hand-crafted shortcuts from a target notion to a place in the textbook that explains it or elaborates on it. As an information retrieval tool, it supports meaningful annotation of the textbook content with manually selected keywords. Finally, as a knowledge organization method, an index is a collection of important domain terms curated by an expert. The fact that index terms and links between them and textbook pages are created manually is underlined. There have been a few research attempts to develop methods and tools for automated index construction [62, 292]; however, their results are far from reliable. More recently developed approaches on terminology extraction either require very large corpora [300] or utilize supervised methods [297]. Finally, existing commercial indexing software such as CINDEX¹ or Index Manager² can help automate some steps of index creation (such as creating a word list, or alphabetizing of an index), but still require manual supervision. At the end of the day, index development is a task that has to be done manually and cannot be done just by anyone. Multiple books (e.g., [17, 34, 145]) and guidelines (e.g., [29, 263]) are written to direct the index creation process. Dedicated associations (e.g., The American Society for Indexing³ and The Society of Indexers⁴) develop and disseminate book-indexing methods and practices. Methodologies for indexing stipulate index length and style, good and bad candidates for index terms, how to maintain consistency when creating hierarchical indices, interrelationships among terms, etc. Usually, it is either a textbook author or a dedicated human indexer who creates the index. As a result, a typical textbook index is not just a collection of words but also a reference model produced by an expert according to a predefined set of rules. Each entry in this model is accompanied with one or more links to relevant textbook pages. Moreover, these pages do not simply mention index entries but provide meaningful references by either introducing corresponding terms or elaborating them.

The previous chapters (🔗 Chapters 3 and 4) described a procedure for the automated extraction of knowledge models from PDF textbooks based on their structure, formatting, and organization patterns. The approach pays special attention to the index sections of the processed textbooks as the source of fine-grained domain terminology and text annotations. The index terms are automatically linked to their corresponding entities in DBpedia, thus enriching the textbook models with additional semantic information and connecting them to the open linked data cloud.

At the same time, even a surface analysis of a typical textbook index can show that not all index entries are equally representative of the domain of the textbook⁵. From an epistemological perspective, an index of any document reflects not only the

¹<http://www.indexres.com>

²<http://index-manager.net>

³<https://www.asindexing.org/>

⁴<https://www.indexers.org.uk/>

⁵E.g., while analyzing textbooks on statistics, it has been observed a rather stable ratio of about 2/3 of all index entries categorizable as relevant to the domain of statistics.

expertise and efforts of its creator, but also the needs of a group of users for whom the index is created, and the task that these users are engaged in [129]. Hence, on the one hand, it can never be expected for a textbook index to represent a fully objective and neutral model with a consistent granularity of cohesive terms covering the domain of the textbook exclusively, completely, and unambiguously. At the same time, it can be expected that the purpose of an index follows the purpose of the textbook itself, namely, to present important notions that help readers better understand a certain subject. Many of the index entries will introduce the core concepts from the target domain, yet there will also be terms included in the index to represent the unique view of the textbooks author, make connections to relevant domains, and present potential use cases, applications, and examples. In other words, even a good index is likely to include a large number of entries that refer to concepts that are either mildly relevant to the target domain, or not relevant at all. For example, in a *statistics* textbook, the terms *mean* and *hypotheses testing* will belong to the core of the main subject. There will also be other, more niche statistical terms, such as *five-number summary* and *cross-validation*. Terms like *factorial* and *De Morgan's laws* are likely to be present as well, yet they are associated with domains related to *statistics*: *mathematics* and *set theory*, respectively. Finally, terms like *Euro coin* and *Bovine Spongiform Encephalopathy* are from entirely different domains, included to enrich the textbook with examples.

This means that the extracted knowledge models from textbooks contain concepts with low domain specificity. This can seriously reduce the value of such models as intelligent services built on top of them would have a hard time distinguishing between relevant and irrelevant domain knowledge. For example, an adaptive learning environment [128] using such a model could misjudge the importance of a certain concept and mistakenly guide students to irrelevant educational material.

Since manual assessment of domain specificity of large knowledge graphs is a time-consuming and a complex task [183], this chapter focuses on developing a method for automated analysis of index terms and identification of their relevance to the domain of a textbook. The information extracted from textbooks and DBpedia is integrated. Textbooks supply index terms and referenced text fragments, while DBpedia provides structural (categories and links) and textual (abstracts) information associated with the entities linked to the index terms.

The contributions of this chapter are two-fold: (1) an approach to identifying the domain specificity relations of index terms; and (2) an evaluation of the accuracy and applicability of the proposed approach.

The rest of the chapter is organized as follows. Section 5.2 presents related work. Preliminary concepts are explained in Section 5.3. The proposed approach is described in Section 5.4. Section 5.5 presents and discusses the evaluation. Finally, Section 5.6 concludes.

5.2 Related Work

5.2.1 Textbook Indices

Index sections are a source of document and domain-specific terms. Surprisingly, textbook indices have not yet received much attention in the literature. NLP tools and heuristic reasoning were applied to extract terminology using the index section of a single book [165]. A security textbook was used as a source of terms to develop a cybersecurity ontology [280]. Besides using index terms, terminology extraction methods can automatically identify and extract core vocabulary of a specialized domain in un- and semi-structured corpora [82, 171].

5.2.2 Domain Specificity

Martín-Moncunill et al. [183] presented a methodology to manually assess the specificity of a large terminology using quantitative and systematic analyses in conjunction with a domain-expert evaluation to classify the terms as "specific" or "not specific" to the studied domain. Their principal findings are that manual domain specificity assessment is time-consuming and complex, and it is necessary to find automatic mechanisms to evaluate large terminologies' domain specificity. Several automatic approaches have been proposed for domain specificity classification. Kida et al. [153] used a sample of terms to query documents from the web and estimate the domain specificity according to the distribution of the domain of those documents. Each target term is classified as one of three possible levels: mostly appears in the domain, generally appears in the domain, and generally does not appear in the domain. A related task is term classification, where terms are classified into possible domains. Rigutini et al. [229] trained a classifier using a set of context terms extracted from the snippets of pages returned by a search engine to assign to a term one category from a predefined set of classes. Gaizauskas et al. [105] first used a terminological service where terms are annotated with domains to create domain vectors, which are then compared against vector representations of documents to classify them into one category. Finally, terms are assigned to the domains of the documents where they appear.

5.2.3 Domain Discovery

An indirect way to classify in-domain terms is to generate a list with relevant terms in a specific domain. Xu et al. [294] applied a TFIDF-based single-word term classifier over a set of already classified documents to extract domain-relevant terms. Milne et al. [196] demonstrated that Wikipedia could be used to extract domain-specific terms/thesauri. Each article in Wikipedia describes a single concept; its title is a well-formed phrase that resembles a term. Additionally, Wikipedia handles synonymy and polysemy, hierarchical categorization, and associative relations (hyperlinks). Also, using Wikipedia, both Vivaldi & Rodríguez [278] and Mirylenka et al. [200] constructed a set of relevant categories to a domain by discovering sub-categories a breadth-first traversal of the category graph, starting from a root category that represents the domain of interest. The former pruned irrelevant en-

tities using an iterative approach and constraints based on already selected parent categories. The latter marked sub-categories as relevant using a classifier trained using the depth, title, and parent categories. Finally, Lalithsena et al. [162] presented an approach to extract a domain-specific hierarchical subgraph of categories and entities in DBpedia. From a set of domain entities as input, the graph is expanded to in-domain categories identified using three types of features: type, lexical, and structural semantics.

5.2.4 Semantic Relatedness

When computing domain specificity, the notion of semantic relatedness is important to identify if an entity (or set of entities) is part of the target domain. Several methods have been proposed to compute the relatedness between two elements. Witten & Milne [289] computed the semantic relatedness of two terms using Wikipedia's network of inter-article links, rather than using the category hierarchy or the textual content of articles. Leal et al. [166] defined a metric that measures relatedness based on the proximity of two terms in DBpedia. Finally, Piao et al. [220] improved the LDS similarity measure [217] by taking into account the similarity of the properties of two entities as well as satisfying fundamental axioms for similarity. At the domain level, Di Noia et al. [73] used a combination of three features (graph-based, text-based, and web-based) to rank DBpedia entities based on their relatedness to one specific domain. Lastly, Ni et al. [210] computed a similarity measure between two concept graphs from different documents. The path scoring methods apply the notion of semantic relatedness to identify if an entity is related to the domain, to other domains, or unrelated.

5.3 Preliminaries

5.3.1 Domain Specificity

Generally speaking, domain specificity categorization refers to the task of assigning to a term the label *used* or *not used* concerning a domain D of interest [153]. This chapter extends the traditional classification into a set L of four domain specificity labels to annotate the index terms. Each label $l \in L$ is one of the following:

- *core-domain*: key index terms that represent the most important and frequently used concepts in D ;
- *in-domain*: additional index terms that belong to D ;
- *related-domain*: index terms from domains related to D ;
- *out-of-domain*: index terms not related to D (often used for pedagogical reasons, e.g., examples, use-cases, summaries).

An additional label, *in-domain+*, is used to group *core-domain* and *in-domain* terms.

5.3.2 DBpedia

DBpedia is a knowledge graph [30] extracted from Wikipedia [22]. Each Wikipedia entry/page is represented as a DBpedia entity. Currently, its English version describes over 6 million entities, uniquely identified by URIs. Knowledge in DBpedia can be queried through its SPARQL endpoint or downloaded as a full RDF model. Each DBpedia entity has an abstract, a category, and a set of links to other entities. Abstracts are extracted from the texts of Wikipedia pages preceding tables of contents. Categories are special kind of entities used to classify and group regular entities on similar subjects. Each entity has one or more categories associated, and each category has a set of sub-categories and super-categories. The symbols \subseteq^c and \supseteq^c are used to indicate that a category is a sub-category or a super-category, respectively. For example, *dbc:Statistics* \subseteq^c *dbc:Probability_and_statistics* and *dbc:Statistics* \supseteq^c *dbc:Applied_statistics*. Also, when \subseteq^c and \supseteq^c are used between an entity and a category, they show the direct category of the entity. For example, *dbr:Mean* \subseteq^c *dbc:Means*. This allows to navigate the non-strict hierarchy of categories using a number of hops (n-hops) to connect categories and entities. Finally, each entity is associated with other entities using the hyperlinks between the corresponding Wikipedia pages of the entities. Figure 5.1 shows five DBpedia entities and how they are connected to *dbc:Statistics* using the categorization system.

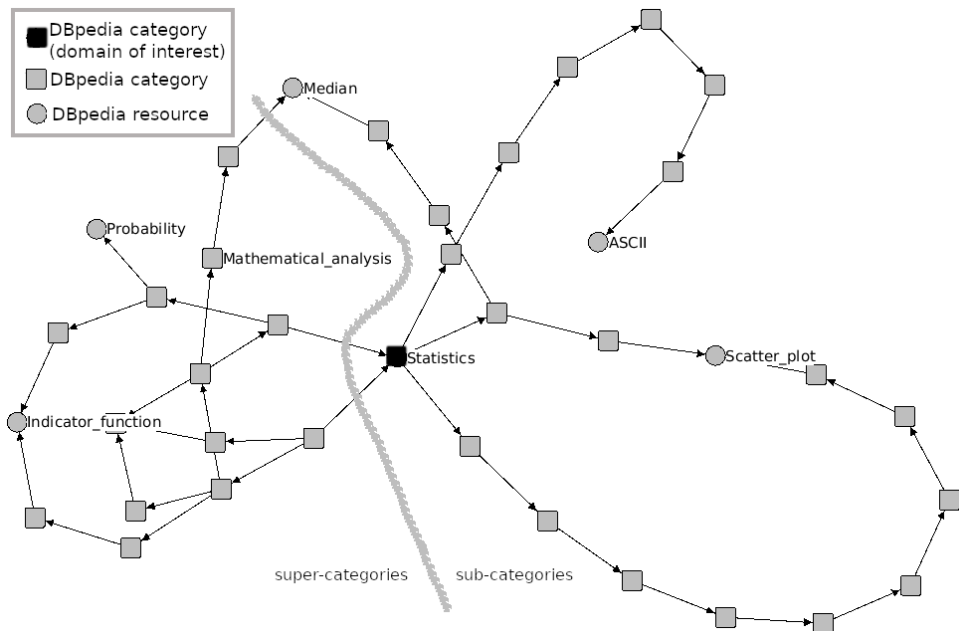


Figure 5.1: Entities connected using the categorization system.

5.3.3 Knowledge Model Extraction and Linking to DBpedia

This thesis has presented a method for the automated extraction of knowledge models from textbooks (☞ Chapters 3 and 4). During the first phase, a knowledge model is extracted from a textbook using its formatting, structure, and organization. Then, the model is enriched with additional semantic information from DBpedia by linking identified index terms into DBpedia entities. The second stage is more relevance for this chapter, therefore a more detailed description of it is provided.

First, the index section of the textbook is processed, all terms are extracted and added to a glossary. Then, during the term linking step, each term is queried against DBpedia. When a result is retrieved, it can be either (1) an entity or (2) a list of candidate entities with the same or similar names. Suppose the result is a single entity r , and it belongs to a sub-category that is at most 3-hops away from the target category (representing D). r is considered to belong to D unambiguously and is linked to the target term. After all index terms have been tossed to the DBpedia SPARQL interface as queries, a set of entities called *SEED* is obtained. Each $r \in SEED$ represents a *seed entity* in D . When multiple entities are retrieved, a similarity score is computed between each candidate entity's abstract and the cumulative abstract of *SEED*. The candidate entity with the highest similarity, which is also above a threshold, is linked to the target term. This second set of entities is called *SLCT*. In the final step, each linked term is enriched with semantic information extracted from DBpedia.

Once the model is fully linked and enriched, it is serialized as an XML file using the TEI. Up to this point, this thesis' approach considers all the extracted index terms equally important for the subject of the textbook, i.e., for the domain model extracted from it. The following section describes a deeper analysis of the textbook indices and refines extracted models by labeling their concepts according to their domain specificity.

5.4 Approach

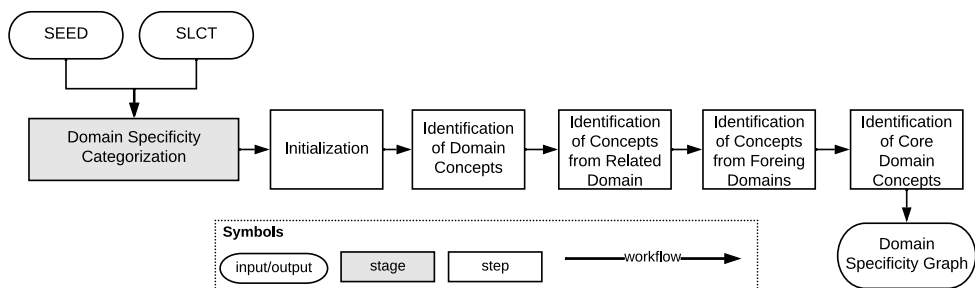


Figure 5.2: Outline of the implemented approach (categorization phase).

To decide whether an entity (and an index term linked to it) belongs to a domain, the proposed approach heavily relies on the structure of the DBpedia category graph that provides an easily navigatable web of (sub)domains for any root domain cate-

gory. However, some properties of this graph present challenges. For example, it is possible for an entity to be connected to a root category using both sub- and super-categories. The former connection denotes belonging to the main domain, and the latter indicates that the entity is related indirectly through a different domain. When both connections exist, it is not possible to discern which one is stronger. In Figure 5.1, *dbr:Median* is connected to *dbc:Statistics* both directly to the root and indirectly through *dbc:Mathematical_analysis*. Another important consideration is that a presence of a direct hierarchical path from a domain category to an entity is not enough to assume that the entity belongs to the domain. In Figure 5.1, *dbr:ASCII* is connected to *dbc:Statistics* by a chain of 6 sub-categories, yet it is not a statistical concept. The third challenge is to decide if entities connected only through super-categories are sufficiently relevant to be considered as a part of a related domain or not (e.g., in Figure 5.1 do entities *dbr:Probability* and *dbr:Indicator_function* belong to domains related to *dbc:Statistics*?). Finally, an entity could have multiple paths to a domain using different categories with varying degrees of relevance, making it necessary to identify the most related one. For example, *dbr:Scatter_plot* has two direct paths to *dbc:Statistics* using only sub-categories in Figure 5.1. As exemplified by Mirylenka et al. [200], it is possible to go from *dbc:Computing* to *dbc:Buddhism_in_China* in just 9-hops.

To combat these challenges, information from textbook models is combined with the content and structural properties of the DBpedia knowledge graph. Figure 5.2 shows the overall process for identifying domain specificity of individual DBpedia entities / index terms. The final result is the refined model, which is essentially a domain specificity graph where each individual vertex represents a concept and each concept is marked with an individual specificity label. It is essential to mention that each domain specificity graph indicates only the specificity of concepts extracted from a source textbook (or a set of textbooks) regarding the target domain; it does not try to label the specificity of all possible entities in DBpedia regarding this domain. The approach contains one stage with five steps in total. The following subsections explain each of the proposed steps. Additionally, the algorithmic representations of the main methods of the approach are provided.

5.4.1 Initialization

The first step of the approach is preparatory. The DBpedia entities matched to the textbook's index terms are divided into two sets: *SEED* and *SLCT* (🔗 Section 5.3.3). An empty domain specificity graph is created. The graph is denoted by $DSG = (V, E)$, where V is a set of vertices representing concepts and categories, and $E \subseteq V \times V$ is a set of unweighted and directed edges representing the hierarchical relations of entities in the categorization system of DBpedia. There are one special vertex, $root \in V$, and two subsets of vertices, $C \subseteq V$ and $CAT \subseteq V$. *Root* represents the main DBpedia category of D (e.g., *dbc:Statistics*). C is composed of **concepts**—DBpedia entities linked to index terms whose domain specificity has been identified. Finally, CAT vertices are DBpedia categories with an identified domain specificity and linked to the concepts in C . Elements in C and CAT are annotated with one of the domain specificity labels from L . A path denoted by $p_i = (root \supseteq^c cat_1 \supseteq^c \dots \supseteq^c cat_n \supseteq^c c) \mid cat_x \in CAT, c \in C, \text{ and } i \in \mathbb{N}$, connects

c to $root$. $P_c = \{p_1, \dots, p_g\}$ is a set of g paths connecting c to $root$. For a p_i , if $\forall cat_x \in p_i, root \sqsupseteq^c cat_x$, c is a *core-domain* or *in-domain* concept. On the contrary, if $\exists cat_x \in p_i, root \sqsubset^c cat_x$, c is a *related-domain* concept. Finally, if $\nexists p_i \mid c \in p_i$, c is an *out-of-domain* concept and it is disconnected from DSG . Figure 5.3 shows a fragment of the domain specificity graph used in Section 5.4.7.

Entities in *SEED* and *SLCT* are called concepts in the next steps since their domain specificity is being determined.

5.4.2 Identification of Domain Concepts

The second step of the approach identifies concepts that belong to the main domain of the textbook. First, concepts from *SEED* are added to the *DSG*. All $c \in SEED$ are considered part of D because they have at least one category three or fewer hops from the $root$. This number of hops was selected after running several experiments. It is also in line with previous work [73, 200]. For each $c \in SEED$, the **path finder** method discovers P_c . This method is a depth-first search [60, chap. 22.3] algorithm that starts with all direct categories of c to find all sequences of form $c \sqsubset^c cat_n \sqsubset^c \dots \sqsubset^c cat_1 \sqsubset^c root$. A path p_i is constructed and added to P_c when the $root$ is found within the maximum number of hoops (3) in a sequence. A graph traversal algorithm is necessary because although SPARQL 1.1 supports property paths [243], it does not return arbitrary paths of variable length. Instead, in practice, it only allows testing reachability [66, 241]. Additionally, even though there are proposed frameworks to overcome the limitations of SPARQL 1.1 [126, 241], the approach needs a method that finds directed paths between different categories in DBpedia taking into account both sub- and super-categories within a maximum distance. The definition of a new query model is not in the scope of this chapter.

Then, the **path scoring** method (Algorithm 1) assesses each $p_i \in P_c$ to assign to each category in the path a score (s_{cat_n}) according to their belonging to D . Each category in a p_i is scored as the average of three sub-scores: s_1 —the similarity between all the category's entities and *SEED*; s_2 —the percentage of category's entities that have a direct link to $root$; and s_3 —the similarity between the category and its super-category along the path. The final score of a p_i is the score of cat_n , which is the direct parent category of c . It is important to note that the same category will have different scores in different p_i since each category score incorporates all its super-categories up to the $root$. After the scoring is finished, the p_i with the highest score is selected. Finally, c is added to C with $l = in-domain$. The categories in the selected p_i are added to CAT with $l = in-domain$. All pairs of vertices in p_i are added to E . By selecting the p_i with the highest score, only the most relevant categories to D are added initially to *DSC*.

In the second part of the current step, more *in-domain* entities are discovered. First, since at this point CAT contains only *in-domain* categories, it is possible to directly add the concepts that belong to any of those categories. Each $c \in SLCT$ is added to C if: $c \sqsubset^c cat_n \in CAT$. Then, for each of the the remaining $c \in SLCT$, all corresponding p_i are discovered with the **path finder** method. For optimization purposes, in this case, a maximum of six hops is used; this helps to avoid finding too many irrelevant paths. Next, paths are scored using the **path scoring** method. However, a inclusion/exclusion mechanism based on thresholds (represented using

Algorithm 1: Path scoring

```

Input: a  $P_c$ 
for  $p_i \in P_c$  do
  for  $m \leftarrow 1$  to  $n$  do //  $n$  categories in  $p_i$ 
     $R_m \leftarrow \text{getEntitiesFromCategory}(cat_m \in p_i)$ 
     $s_1 \leftarrow \text{cosSim}(\text{getAbstr}(R_m), \text{getAbstr}(SEED))$  // captures the
      sim between the category and  $D$  (represented by  $SEED$ )
     $s_2 \leftarrow \text{countDirectLink}(R_m, root) / |R_m|$  // captures the % of
      entities that directly mention  $D$ 
     $s_3 \leftarrow s_{cat_{m-1}} * \text{cosSim}(\text{getAbstr}(R_m), \text{getAbstr}(R_{m-1}))$ 
      // captures the sim between the current and the previous
      category in the path
     $s_{cat_m} \leftarrow \frac{s_1 + s_2 + s_3}{3}$ 
  end
end

```

lower case Greek letters) is added to detect when a category/path deviates too much from D . A p_i is excluded if $s_2 < \alpha \wedge s_3 < \beta$ for cat_n . After discarding, if c has at least one p_i , c is added to C with $l = in\text{-}domain$. The categories in the highest scored p_i are added to CAT with $l = in\text{-}domain$. All pairs of vertices in p_i are added to E .

5.4.3 Identification of Concepts from Related Domains

The third step of the approach is to identify the concepts that are not a part of D , but still relevant enough to be considered from related domains. First, for each $c \mid c \in SLCT \wedge c \notin C$, all the p_i that connect c to $root$ indirectly are discovered with the **related path finder** method. This method is similar to **path finder**, but it goes up (using super-categories) and down (using sub-categories) along the hierarchy of categories, which allows finding paths of the form: $p_i = (root \sqsubseteq^c \dots \sqsubseteq^c cat_x^{sp} \supseteq^c \dots \supseteq^c cat_n \supseteq^c c)$, which indirectly connect c to the $root$ using a super-parent category cat_x^{sp} . In practice, this method finds the lowest common ancestors of c and $root$ [28]. For optimization purposes, and due to the fact that using super-categories might result in two very unrelated domains to be connected (e.g., *dbc:Statistics* and *dbc:Musicology* are connected through the *dbc:Academic_disciplines* super-parent category), a maximum of eight hops is used.

Next, indirect p_i s are scored using the **related path scoring** method with an exclusion threshold. This method is similar to **path scoring**, with only one change in s_2 : it checks the intersection between the links from the category's entities and the links from the $SEED$ entities. It has been shown that entities sharing similar links are related [261]. Using the exclusion mechanism, a p_i is too unrelated to D if $s_2 < \gamma \vee s_{cat_n} < \delta$ for cat_n . After discarding, if a concept has at least one indirect p_i remaining, c is added to C with $l = related\text{-}domain$. The categories and vertices in the best p_i are added to CAT with $l = related\text{-}domain$ and to E , respectively.

In rare cases, some concepts can be considered *in-domain* even though there is no direct path to $root$. For example, the *dbr:Sample_space* concept belongs to the

probability domain, but it is a significant concept of statistics. For such cases, the **related path assessment** method (Algorithm 2) checks five constraints to identify an entity as *in-domain* despite an indirect p_i . First, if the related p_i is connected through a sibling category of the *root*. Second, if the score of cat_n is high enough ($> \epsilon$). Third, if the percentage of shared links is high enough ($> \zeta$). Fourth, if the similarity between the entity and *SEED* is high enough ($> \eta$). Fifth, if the entity and the *root* link to each other. If at least four out of five constraints are met, the l associated to c is changed to *in-domain*.

Algorithm 2: Related path assessment

Input: a p_i
 $cstr_1 \leftarrow$ **if** p_i of the form: $root \sqsubseteq^c cat_1 \supseteq^c cat_2 \supseteq^c \dots \supseteq^c c$ **then** 1 // c is connected through a sibling category of *root*
 $cstr_2 \leftarrow$ **if** $s_{cat_n} > \epsilon$ **then** 1 // the score of the path is high
 $cstr_3 \leftarrow$ **if** s_2 of $cat_n > \zeta$ **then** 1 // the % of shared links between c and D is high
 $cstr_4 \leftarrow$ **if** $cosineSim(getAbstr(c), getAbstr(SEED)) > \eta$ **then** 1 // the similarity between c and D is high
 $cstr_5 \leftarrow$ **if** $countDirectLink(c, root) \vee countDirectLink(root, c)$ **then** 1 // c and *root* link to each other
 $cstr_t \leftarrow \sum_{n=1}^5 cstr_n$

5.4.4 Identification of Concepts from Foreign Domains

The fourth step of the approach identifies the entities that are from domains that are not related to the subject of the textbook. This process is straightforward. Any remaining $c \mid c \in SLCT \wedge c \notin C$ is unrelated to D and is added to C with $l = out-of-domain$.

5.4.5 Identification of Core Domain Concepts

The final step evaluates all $c \in C$ with $l = in-domain$ to see if any of them represent one of the most important concepts in D , which are the *core-domain* concepts. The assumption is that the most used entities through textbooks and DBpedia should indicate the most relevant concepts in D . The **core concepts assessment** method (Algorithm 3) is applied to discover such concepts. First, a popularity score is assigned to each concept $c \in C$ based on how many times it has been seen in the textbooks and the number of other concepts linking to it in the *DSG*. The entities with the popularity scores in the upper quartile are selected. Finally, only the concepts that are referenced widely from outside D are selected. It is considered that core concepts are so relevant that they are also used in other domains. If most of the entities that reference (link) the concept belong to a different domain, then it is marked as a *core-domain* concept. To check if an entity res belongs to D or a different domain, a simplified version of the *related path scoring* method is used. Each of the direct categories of an entity is checked: if the combined score of the similarity

Algorithm 3: Core concepts assessment

```

Input: a  $p_i$ 
 $core \leftarrow \emptyset$ 
for  $c \in C \wedge l_c = \text{in-domain do}$ 
   $pop_1 \leftarrow \frac{\sum_{t=1}^{\#textbooks} 1 \text{ if } c \in textbook_t}{\#textbooks}$ 
   $pop_2 \leftarrow \frac{|\{e: e \in E \wedge e = \{x, c\}\}|}{\max\{|\{e: e \in E \wedge e = \{x, c'\}\}| : \forall c' \in C\}}$ 
   $pop_c \leftarrow \frac{pop_1 + pop_2}{2}$ 
end
for  $c \in C \wedge pop_c \in \text{upper quartile do}$ 
   $general \leftarrow 0, specific \leftarrow 0$ 
   $LINKS \leftarrow \text{entities that link to } c$ 
  for  $res \in LINKS \text{ do}$ 
    for  $cat \mid cat \supseteq^c res \text{ do}$ 
       $R_{cat} \leftarrow \text{getEntitiesFromCategory}(cat)$ 
       $abstractsR_{cat} \leftarrow \text{getAbstr}(R_{cat})$ 
       $abstractsSEED \leftarrow \text{getAbstr}(SEED)$ 
       $sim \leftarrow \text{cosSim}(abstractsR_{cat}, abstractsSEED)$ 
       $links \leftarrow \frac{|\text{getLinks}(R_{cat}) \cap \text{getLinks}(SEED)|}{|\text{getLinks}(R_{cat})|}$ 
       $score \leftarrow \frac{sim + links}{2}$ 
      if  $score > \theta$  then
         $specific \leftarrow specific + 1$ 
      else
         $general \leftarrow general + 1$ 
      end
    end
  end
  if  $general > specific$  then
     $coreTerms \leftarrow coreTerms \cup \{c\}$ 
  end
end

```

between entities in the category and the percentage of shared links between the *seed entities* is below θ , the category is considered to be from a different domain. Finally, if most of the categories from r are from a different domain, then $r \notin D$.

5.4.6 Thresholds

Table 5.1: Threshold values.

Threshold	α	β	γ	δ	ϵ	ζ	η	θ
Value	0.1	0.4	0.3	0.25	0.3	0.5	0.2	0.1

After several experiments, the used thresholds were calibrated with the values

shown in Table 5.1. The selected values were flexible but also robust enough to achieve good results in two very different domains: *statistics* and *ancient philosophy* (🔗 Section 5.5). Section 5.5.4 includes a brief discussion on threshold calibration.

5.4.7 Example

Figure 5.3 presents the domain specificity graph of one *statistics* textbook [69]. In miniature, the whole graph is presented. The subgraph in the center is a zoom in and it contains the same five entities as in Figure 5.1, but the two graphs (named DOMAIN and DBPEDIA respectively) look completely different. In the DOMAIN graph, each entity is a concept with a clear relation to the domain of interest; there are no multiple paths from different domains to the entities, as in the DBPEDIA graph. The challenges described at the beginning of this section have been addressed. The *dbr:Median* concept had two possible paths and relations in the DBPEDIA graph, but in the DOMAIN graph it has been identified as a concept belonging to the domain, even as *core-domain*. There is a connecting path from *dbc:Statistics* to *dbr:ASCII* in DBPEDIA, but in DOMAIN, it has been identified as a *out-of-domain* concept. The *dbr:Probability* and *dbr:Indicator_functions* have been considered relevant enough to be classified as *related-domain* concepts in the DOMAIN graph. Finally, the most relevant path from *dbc:Statistics* to *dbr:Scatter_plot* has been identified in DOMAIN, in contrast, there were two possible paths in the DBPEDIA graph.

The domain specificity graphs identify not only the type of domain specificity relationships, but also allow for explaining why the relations exist. For example, in the DOMAIN graph, *dbr:Indicator_function* is a related concept to *dbc:Statistics* from the *dbc:Probability* domain.

5.5 Evaluation

Three evaluation experiments have been conducted using a local copy of DBpedia (version 2020.12.01). The ground truth models generated for the first two evaluations are made publicly available⁶.

5.5.1 Evaluation One: MAIN DOMAIN

The goal of the first experiment was to examine how well the proposed approach can identify concepts that belong to a domain. This evaluation uses textbooks about *statistics*. *Statistics* is richly connected to many other domains (different sub-fields of mathematics and computer science). For this reason, creation of a comprehensive list of *related-domain* entities is not practically feasible. Therefore, this evaluation was interested only in the classification accuracy of the *core-domain* and *in-domain* terms.

Datasets & Procedure

Ten introductory statistics textbooks were used [64, 69, 72, 76, 90, 101, 144, 174, 272, 282]. First, for each textbook, a knowledge model was extracted and enriched

⁶<https://github.com/intextbooks/domain-specificity>

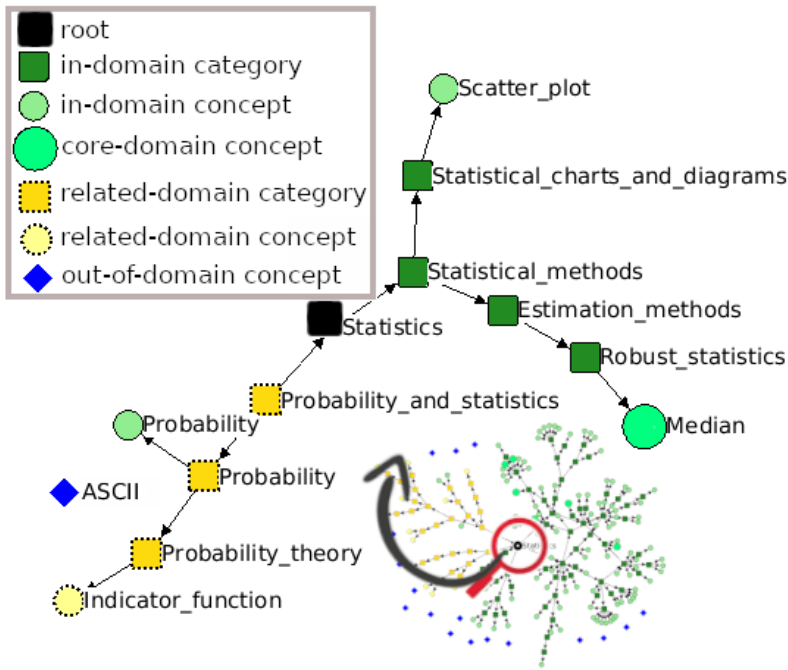


Figure 5.3: Example of a domain specificity graph.

with semantic information from DBpedia using the approach presented in this thesis (🔗 Chapters 3 and 4). For the enrichment process, the DBpedia category *dbc:Statistics* was used. Then, the described approach for the domain specificity graph construction was applied. The approach was executed over the combination of all selected textbooks to obtain their cumulative knowledge graph. To determine the true domain specificity label of all concepts, a ground truth was created using multiple sources. First, the *core-domain* concepts were identified by checking the intersection of six statistical glossaries ([83, 100, 138, 163, 256, 274]). A term was marked as *core-domain* if it appeared in at least two glossaries or in Landrum [163], which is a master list of core statistical terms created manually and rated by statistics instructors. After finding the corresponding DBpedia entities, 186 core concepts were obtained. Then, the *in-domain* concepts were identified by merging three sources: (1) all DBpedia entities extracted from a list of statistical articles in Wikipedia⁷; (2) a list of statistical DBpedia entities constructed using the ISI Glossary (🔗 Chapter 4, Section 4.4.3); and (3) all entities in DBpedia which have explicitly encoded in their URI that they belong to statistics (e.g., *dbp:Q-value_(statistics)*). After removing the concepts already marked as core, a final list of 2658 *in-domain* concepts was obtained. Any concept outside the ground truth was classified as *other-domain* (meaning it can be either *related-domain* or *out-of-domain*). The confusion matrix was calculated and the standard accuracy, precision,

⁷https://en.wikipedia.org/wiki/List_of_statistics_articles

and recall were used as the evaluation metrics [214, chap. 9]. A simple path-based baseline (BL) was also used for comparison, as in previous works [162, 200]. BL only used the DBpedia categorization system: if an entity could be reached from the domain root using sub-categories within 2-hops, it was classified as *core-domain*; using sub-categories within 4-hops it was classified as *in-domain*; otherwise, it was considered being too far from the domain root and was classified as *other-domain*. Finally, the McNemar's test [240] (a non-parametric test for paired nominal data) was used to analyze the statistical significance of differences in accuracy between the graph and the baseline.

Results

Table 5.2: Results for domain boundary detection (statistics).

	<i>n</i> = 648	actual relation		Σ	Precision	Recall	Accuracy
		in-domain	other-domain				
GRAPH	in-domain	383	11	394	.972	.897	.915*
	other-domain	44	210	254	.827	.950	
	Σ	427	221				
BL	in-domain	362	13	375	.965	.848	.880
	other-domain	65	208	273	.762	.941	
	Σ	427	221				

*Statistical significance against BL.

Table 5.3: Results for core domain boundary detection (statistics).

	<i>n</i> = 648	actual relation			Σ	Precision	Recall	Accuracy
		core-domain	in-domain	other-domain				
GRAPH	core-domain	45	5	0	50	.900	.306	.762*
	in-domain	94	239	11	344	.695	.854	
	other-domain	8	36	210	254	.827	.950	
	Σ	147	280	221				
BL	core-domain	59	87	6	152	.388	.401	.637
	in-domain	70	146	7	223	.655	.521	
	other-domain	18	47	208	273	.762	.941	
	Σ	147	280	221				

*Statistical significance against BL.

Table 5.2 shows the results of focusing on the boundary of the target domain (is a concept a part of the domain or not). The approach outperforms the baseline in terms of precision, recall and accuracy. The later difference is statistically significant as indicated by the McNemar's test ($\chi^2(1, N = 648) = 13.829, p < 0.001$).

When the task becomes more complex, and the approach tries to find which concepts belong to the core of the domain, the gap between the baseline and the proposed approach increases overall, as seen in Table 5.3. The baseline outperforms the proposed approach only in terms of recall of *core-domain* concepts. However, on the other two recall values as well as the precision of obtained labels, this chapter's approach clearly performs better. It is important to mention that for many tasks, precision is a much more important metric as its increase leads to elimination of type I error. It is also worth noticing, that the proposed approach makes consid-

erably fewer serious mistakes (labeling *core-domain* concepts as *other-domain* and vice versa). When it comes to accuracy, again, the difference between the two methods was significant according to a McNemar’s test ($\chi^2(1, N = 648) = 35.359, p < 0.001$).

5.5.2 Evaluation Two: MULTIPLE DOMAINS

This evaluation has analyzed the cross-domain robustness of the approach by applying it to the *ancient philosophy* domain. Additionally, it has checked the ability of the approach to distinguish between *in-domain*, *related-domain*, and *out-of-domain* concepts.

Datasets & Procedure

One textbook about *ancient Greek and Roman philosophy* [150] was used. Its knowledge model was extracted and enriched with corresponding DBpedia entities using the *dbc:Ancient_Greek_philosophy* category, after which its domain specificity graph was generated. To determine the true domain specificity label of the concepts, a ground truth was created manually. For each concept, one of the three possible labels (*in-domain*, *related-domain*, and *out-of-domain*) was assigned. The concepts directly associated with the "topics" inferred from the chapters and covered across the textbook (e.g., *Plato*) were considered as *in-domain*. *Related-domain* concepts corresponded to general philosophical notions and philosophers, people and places from the same era, and auxiliary philosophical terms (e.g., *logic*). General and broad concepts (e.g., *art*) were classified as *out-of-domain*. In case of doubt, the following resources were used: the textbook itself, the Stanford Encyclopedia of Philosophy⁸, and general web searches to clarify the relevance of a concept in the domain. A total of 426 unique concepts were classified. The same metrics as in the previous experiment were used. BL was applied as well, but in this case, any entity reached directly within 4-hops was classified as *in-domain*, within 4-hops using an indirect path as *related-domain*, or otherwise as *out-of-domain*. The McNemar’s test was applied to verify statistical significance.

Results

Table 5.4: Results for multi-domain boundary detection (ancient philosophy).

		actual relation			Σ	Precision	Recall	Accuracy
$n = 426$		in-domain	related-domain	out-of-domain				
GRAPH	in-domain	129	8	0	137	.942	.977	.932*
	related-domain	3	148	16	167	.886	.937	
	out-of-domain	0	2	120	122	.984	.882	
	Σ	132	158	136				
BL	in-domain	127	8	2	137	.927	.962	.723
	related-domain	1	51	4	56	.911	.323	
	out-of-domain	4	99	130	233	.558	.956	
	Σ	132	158	136				

*Statistical significance against BL.

Table 5.4 presents the results. They show that the accuracy of the approach remains high and stable in a different domain as well. Also, when identifying the

⁸<https://plato.stanford.edu/index.html>

possible domains of concepts, the accuracy of the approach is 21 points higher than the baseline (93% vs. 72%). The proposed method's combination of content and structural properties gets high precision and recall values across all possible labels compared to the method using only category-based paths (BL). Some entities have both direct and indirect paths to the domain, and the use of a scoring function is the key to decide the proper relation to the main domain. Finally, there is a statistically significant difference between the accuracy of the model and the baseline according to a McNemar's test ($\chi^2(1, N = 426) = 62.959, p < 0.001$).

5.5.3 Evaluation Three: APPLICATION

The goal of this experiment was to show the added value of domain specificity labels. To that end, the proposed approach was applied to model documents for a simple query-based retrieval task.

Datasets & Procedure

Apache Lucene⁹ was employed to construct a web search system for textbooks. The experimental model (*IND*+) used a combination of textual content, index terms, and *in-domain*+ labels to model (sub)chapters of textbooks, where index terms and labeled index terms received more weight. For each search query, a ranked list of documents was retrieved using a standard tf-idf scoring formula [239]. As baselines, two variations of the system were used: *TFIDF* uses only the content, and *IND* uses the content and the index terms (without domain specificity labels). To evaluate the added value of domain specificity information against the baselines, a standard procedure was followed [177, Chapter 8]. Two textbooks from Section 4.1 [69, 174] were selected as the target document collection. A set of queries was composed using 20 syllabi of university-level statistics courses¹⁰ to represent typical information needs of students: 100 queries were selected from statistics syllabi and 40 from statistics-related syllabi. Additionally, ten queries were selected from the ToC from an additional textbook [218]. Three experts in statistics were recruited to generate a set of relevance assessments for query-document pairs. For each query, each rater indicated the relevant chapters and subchapters from the two textbooks using a three-point scale: (1) partially relevant; (2) relevant; and (3) highly relevant. Finally, the experts' assessments were applied as the ground truth to evaluate the results retrieved by the queries using average NDCG at 1, 3, and 5. NDCG@1 measures the effectiveness of retrieving the most relevant document, while @3 and @5 measure it for three and five most relevant documents, respectively. Inter-reliability between raters was calculated using pooled Fleiss' kappa across all queries [67, 211]. The resulting kappa of 0.36 is considered a fair agreement. Given that all raters were experts and made their relevance assessments fully independently from each other, a smoothed factor¹¹ was used to compute the final relevance for each query-document pair. The average relevance was multiplied by 0.76, 0.91, and 1 for a document identified by one, two, or three raters, respectively.

⁹<https://lucene.apache.org/>

¹⁰E.g., <https://www.stat.berkeley.edu/~mgoldman/sylsm09.pdf>

¹¹Calculated using $1 + ((1 - support) \cdot (\log(support) \cdot support))$, which was inspired by the *expected information gain* formula used in decision trees.

Table 5.5: Retrieval of documents.

Metric / Model	TFIDF	IND	IND+
NDCG@1	M	.303	.410
	SD	.386	.411
	t	-3.24	.639
	df	140	140
	Sig.	< .01	.524
NDCG@3	M	.327	.427
	SD	.349	.355
	t	-4.74	2.5
	df	140	140
	Sig.	< .0001	< 0.5
NDCG@5	M	.348	.459
	SD	.355	.358
	t	-4.95	2.04
	df	140	140
	Sig.	< .0001	< 0.5

Results

NDCG mean values and standard deviations are presented in Table 5.5. Results show that the *IND+* model using domain specificity of index terms supports more accurate retrieval of relevant documents. Pairwise t-tests confirm that the difference between *IND+* and *TFIDF* is significant across all three metrics, and between *IND+* and *IND* for NDCG @3 and @5 (Table 5.5).

The three systems perform similarly when a query corresponds to a single concept that appears textually in concise (sub)chapters (e.g., "nonparametric statistics"). The use of index terms in both *IND* and *IND+* helps to find synonyms. For example, the "graph" query is matched to concepts like *chart*, *bubble plot*, and *scatter diagram*. Finally, the domain specificity data are useful as it encodes information about the relevance of a term in a domain. The *IND+* model is better than *IND* when the queries correspond to concepts with identified domain specificity (e.g., "type I and type II errors" which is a *core-domain* concept).

IND+ is a simple model that could be further improved using the full potential of domain specificity and the *DSG*. For example, queries like "histograms and other graphs" could be matched to their corresponding main concept in the *DSG* (*db: Histogram*) to use the information from its parent category (*dbc: Statistical_charts_and_diagrams*) to identify related concepts (*dbc: Pie_chart*, *dbc: Box_plot*, ...). Creating a more robust model is part of future work tasks.

Table 5.6: Coverage of *in-domain+* concepts (statistics).

Set size	1	2	3	4	5	6	7	8	9	10
M	97	157	202	239	270	297	321	343	362	383

5.5.4 General Discussion

The proposed approach is sensitive to the incompleteness / inconsistencies in DBpedia. Since classification of entities into categories is done mainly by Wikipedia authors, it is a subjective process and some categories can be missing or inappropriate. For example, the categories of *dbp:Mutual_exclusivity* do not include *probability*, therefore it is not classified as a *related-domain* concept of *statistics*. Some other entities have categories that reflect a falsely high relevance to the domain. In the *ancient philosophy* evaluation, *dbp:Alcibiades* is marked as a *pupil of Socrates* in DBpedia, and therefore it is classified as an *in-domain* concept, while, in fact, he was not a philosopher.

It was also noticed that for the *ancient philosophy* domain, the abstracts of the entities tend to be more general than for *statistics*, and higher thresholds in the scoring functions would have been beneficial. One possible solution for this situation is to automatically adjust the thresholds based on the seed entities' path scores.

Finally, as more textbooks are used in the same domain, the coverage of concepts in the domain increases. The number of *in-domain+* concepts discovered when increasing the number of textbooks was calculated. For the *statistics* domain, ten textbooks were used to explore all possible sets (permutations) when selecting from 1 to 10 textbooks. Table 5.6 contains the average numbers of correctly discovered *in-domain+* concepts for each set.

5.6 Conclusion and Future Work

This chapter explored how textbook indices can be used to extract high-quality knowledge graphs in narrow domains. Specifically, an approach was presented to automatically categorize terms in relation to their relevance to the main subject of a textbook (i.e., domain specificity). The traditional binary classification of specificity was extended into four labels that better reflect the degree of how much a term belongs to the target subject: *core-domain* for the most important main domain concepts, *in-domain* for regular concepts in the main domain, *related-domain* for neighboring domains, and *out-of-domain* for concepts unrelated to the main domain, but important for pedagogical reasons. In practice, the *core-domain* and *in-domain* terms identify the central area of the main domain, while the *related-domain* identify adjacent areas. Ultimately, this approach allows addressing one of the biggest problems of textbook indices as sources of domain knowledge, namely the presence of a large portion of entries that are either weakly related or unrelated to the main domain.

The evaluations experiments have demonstrated that the approach is capable to distinguish with high accuracy between the concepts relevant and non-relevant to the domain. Additionally, the accuracy of identifying the most important *core-domain* concepts also remains considerably high. Moreover, the approach has been successfully tested across two different domains (*statistics* and *ancient philosophy*). Finally, it was showed that the domain specificity information can be helpful in the context of information retrieval tasks.

The next step is to further experiment with the potential of domain specificity information when using the knowledge graphs in combination with powerful language

models. One possible direction is to explore informed word embeddings [266, 299], where the index terms and different weights according to their domain specificity could be used to produce embeddings reflecting a domain of interest.

The Power of the Curve:
Measuring the Quality of
Extracted Concepts Using
Learning Curve Analysis

Abstract Evaluating the quality of the extracted knowledge representation (concepts) for modeling tasks (e.g., domain and student modeling) is essential to ensure that they are valid knowledge units. This chapter describes an experiment using learning curve analysis to evaluate textbook concepts' cognitive validity and granularity. The error rate of students' interaction with annotated learning activities was plotted to determine if the learning concerning the concepts of interest follows the power law (i.e., there is a learning process). The results show that the evaluated concepts are meaningful knowledge components for modeling. Additionally, concepts extracted from textbooks provide different levels of granularity. Ultimately, the evaluation presented in this chapter demonstrates the knowledge modeling quality of concepts extracted from textbooks.

A modified version of this chapter has been submitted for publication:

Alpizar-Chacon, I. & Sosnovsky, S., "Measuring the quality of domain models extracted from textbooks with learning curves analysis", *In submission*, n.d.

6.1 Introduction

The previous chapters have explored the quality of the extracted models in terms of accuracy, semantics, coverage, and specificity. The information in the models corresponds to the one in the textbooks. In addition, the domain terms are enriched with additional semantic information from DBpedia. Moreover, the coverage of the domain can be increased by adding more textbooks. Finally, the concepts that are outside of the domain of the model can be identified. In other words, the shape and boundaries of the domain have been explored. Now, it is time to assess the quality of the extracted concepts.

This chapter explores the validity of the extracted concepts as Knowledge Components (KCs)¹ for knowledge modeling and assessment (cognitive validity). Additionally, the concepts are evaluated to see if they cover too much or too little knowledge (granularity).

The extracted concepts can be validated by borrowing a technique used in the field of AIES. For educational systems, evaluating models is important, specifically, the quality from the point of the domain knowledge representation. For such evaluation, the traditional approach is to use learning curve analysis [182].

Learning Curve Analysis

Learning curves are graphs that plot performance on a task versus the number of attempts to practice. Performance is usually measured using the proportion of incorrect responses (the error rate) for a KC that is relevant during student practicing. Learning curve analysis is used to qualify learning performance. If learning occurs for the KC being measured, the learning curve will follow the power law [209]. That is, the error rate of a KC decreases as the power function of the number of attempts involving this component. As pointed out by Sosnovsky & Brusilovsky, the "learning curves that better approximate the power law correspond to more cognitively valid units of knowledge" [252].

Formula 6.1 presents the power law. T is the performance measurement, and N is the number of attempts. The constant B represents the y-axis intercept, which for learning curves is the performance on the first attempt, i.e., the error rate at $x = 1$. The power law slope (decay factor) is depicted by α and indicates the speed at which performance improves, i.e., the learning rate. α is equivalent to the linear slope using a log-log axis. Additionally, the fit (R^2) of the power law to the data is measured to quantify the evidence of learning.

$$T = BN^{-\alpha} \quad (6.1)$$

The parameters of the power laws are used to give insight into the validity of KCs as units of learning. A positive slope indicates a decreasing curve and, therefore, a learning effect. A high fit indicates that the KC successfully identifies the student's learning.

¹KC: "An acquired unit of cognitive function or structure that can be inferred from performance on a set of related tasks" [158]. This term is often used to refer generally to different units of knowledge, like a schema, production rule, concept, fact, or skill.

Learning curves are generated using student logs as the source of data. A sequence is generated for each student and KC containing the attempts (correct or incorrect) over time. However, individual sequences are not enough data to produce a smooth power law. Therefore, student data are aggregated over all students for each KC to see each trend. Figure 6.1 shows an example of a learning curve.

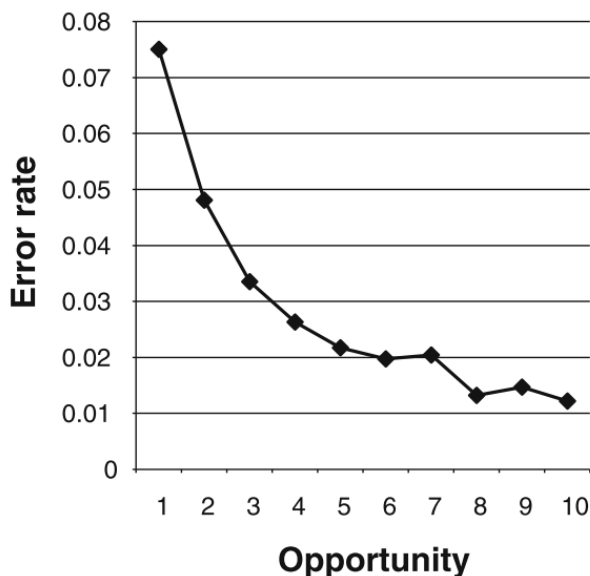


Figure 6.1: An example of a learning curve (image fragment from Martin et al. [182]).

Motivated by the mentioned learning curve analysis, this chapter describes an experiment to assert the cognitive validity and granularity of concepts extracted from textbooks. The performed experiment, in the domain of Python programming, can be summarized as follows. First, a set of concepts was extracted from textbooks using the approach presented in this thesis (📌 Chapters 3, 4, and 5). Then, a set of learning activities was annotated with the extracted concepts to indicate their expected learning outcomes. After that, log data of students interacting with the annotated learning activities were processed, and learning curves were generated for each concept.

The analysis of the learning curves showed that concepts extracted from textbooks are cognitively valid knowledge components for domain knowledge modeling. Additionally, textbooks provide both fine- and coarse-grained concepts, with the former modeling the knowledge being learned better. Finally, as more textbooks are used as the source of concepts, the domain's coverage and diversity increase.

The main elements of this chapter are: (1) an overview of learning curves as a method for evaluating KCs; (2) an experiment to test the cognitive validity and granularity of textbook concepts; and (3) an analysis of the results from multiple

perspectives.

The remainder of this chapter is organized as follows. Related work is described in Section 6.2. Section 6.3 provides the details on the data and procedure of the experiment. Results are presented and analyzed in Section 6.4. Finally, Section 6.5 outlines conclusions and future work.

6.2 Related Work

This section presents studies where learning curves have been used for the evaluation of educational systems.

Anderson et al. [18] used learning curves to compare two sets of production rules in a LISP tutor. The first set contained "old" productions introduced in previous lessons. The second set was formed with "new" productions being introduced in the current lesson. The analyses showed that performance on both sets improved with practice within the lesson.

Martin et al. [181] suggested learning curve analysis as an evaluation approach for Intelligent Tutoring Systems (ITS). Specifically, the slope and the fit of learning curves were used to measure the speed at which the student learns KCs from an underlying domain model. Considerations for evaluating systems with different domain models were also discussed. In a later work, Martin et al. [182] presented three studies demonstrating the learning curves' power to drive changes in the student model. One of the studies focused on the model granularity, showing that more general components tend to produce low-quality learning curves.

Sosnovsky & Brusilovsky [252] used learning curves to evaluate coarse-grained topics as knowledge assessment units in an AIES. The results showed that while topic-based units can be used as knowledge components, they still cover too much learning content compared to finer-grained units. Another finding of the study showed that too difficult or too easy activities affect both learning and knowledge assessment effectiveness.

Rivers et al. [231] applied learning curve analysis to programming data to determine which programming elements students struggle with the most when learning Python. The analyzed KCs were extracted automatically from the Python Abstract Syntax Tree. The analysis gave insight into the KCs that are already learned, not being learned, still being learned, and successfully learned. In this study, the aggregation of KCs was not explored.

Finally, thanks to learning curve and predictive validity analyses, Tacoma et al. [257] identified weaknesses in the student models and instructional modules for a first-year university statistics course. The analysis showed that only half of the evaluated KCs were well-defined.

6.3 Experiment

This experiment aims to analyze the conceptual representation of knowledge extracted from textbooks using the proposed approach in this thesis. First, the validity of concepts as cognitive KCs is explored. Then, the experiment analyzes how

fine-grained the extracted concepts are in the domain. These properties (cognitive validity and granularity) are verified by applying learning curve analysis to quantify the learning performance of the concepts. The rest of this section explains the data and procedure used.

6.3.1 Data

The data used for this experiment are textbooks and learning content. Textbooks are used as the source of concepts. The learning content refers to (1) learning activities annotated using the textbook concepts; and (2) students' interactions with those learning activities. The interactions are extracted from multiple student logs.

As the source of domain knowledge, three introductory Python programming textbooks are used: Introduction to computation and programming using Python (*BOOK#1*) [120], Think Python (*BOOK#2*) [80], and Python for everybody (*BOOK#3*) [244].

The learning content comes from the Mastery Grids (MG) for Python system [116]. MG is an online personalized system offering students in introductory Python courses interactive learning activities to practice their programming skills. Each activity is designed to train one or more KCs related to Python programming. The conceptual model² was automatically derived using the Python parser (similar to Rivers et al. [231]). In MG, there are five types of learning activities; however, only two are considered problem-solving activities with an adequate level of difficulty. Quizzes for Python Educational Testing (QuizPET) are parameterized activities where the students need to write short answers, typically the output of code or the final value of a variable [118]. Parson's problems are a form of coding activities where students construct a program by selecting from a collection of given code fragments [216]. Figures 6.2 and 6.3 present an example of a QuizPET and a Parson's activity used in MG, respectively. Learning activities in MG are grouped into ordered topics. In total, there are 15 topics in MG. Table 6.3 presents the topics in the order they are supposed to be trained.

Eleven datasets of students' interactions with MG are used for this experiment. The University of Pittsburgh provided the datasets using the PSLC DataShop³ [157]. The datasets are obtained from interactions gathered with a similar MG configuration from students enrolled in 11 different instances of Python courses taught from 2019 to 2021 in three different higher education institutions: one in the United States, one in Europe, and one in Australia. The main properties for each interaction are: student id, timestamp, session id, learning activity, duration, and outcome (correct or incorrect).

A final combined dataset containing 57929 interactions of 465 students with 85 (51 QuizPET and 34 Parson's) activities is used in this experiment.

6.3.2 Procedure

Figure 6.4 shows the experiment's procedure. The first step is extracting the domain knowledge from textbooks and selecting the relevant concepts. Then, the learning

²<http://acos.cs.hut.fi/static/python-parser/ontology.png>

³<http://pslcdatashop.web.cmu.edu>

```

var = 4
if var == 0:
    print("Got zero")
elif var % 2 == 0:
    print("Got an even value")
else:
    print("Got an odd value")

```

What is the output?

Be careful of the whitespace(space,newline) in your answer.

Figure 6.2: An example of a QuizPET learning activity in MG.

Drag from here

```

for animal in zoo:
    zoo = ["bear", "lion", "camel"]
    print(animal)

```

Construct your solution here

[New instance](#) [Get feedback](#)

Construct a program that prints out all the animals in the zoo-variable.

Figure 6.3: An example of a Parson's learning activity in MG.

activities are annotated with the selected concepts according to their expected learning outcomes. After that, the interactions from the students are aggregated, filtered out, and augmented. Finally, learning curves are generated for each concept. The rest of this section details each step.

Domain Knowledge Generation

In this first step, a knowledge model is extracted and enriched for each textbook (☛ Chapters 3, and 4). The general *dbc:Computer_programming* category is used to indicate the main domain of the textbooks. Although there is a specific Python category in DBpedia (*dbc:Python_(programming_language)*), it groups entities more related to the development of Python as a language (e.g., *dbc:History_of_Python* and *dbc:CircuitPython*) than programming concepts used in Python (e.g., *Conditional_(computer_programming)*). After the individual models are generated, the domain knowledge from the three models is combined into a single model (*PYTHON*). Repeated domain terms across the three textbooks are merged in the combined model (☛ Chapter 4, Section 4.3.5 provides a detailed explanation on the identification of repeated terms). Finally, the terms from the

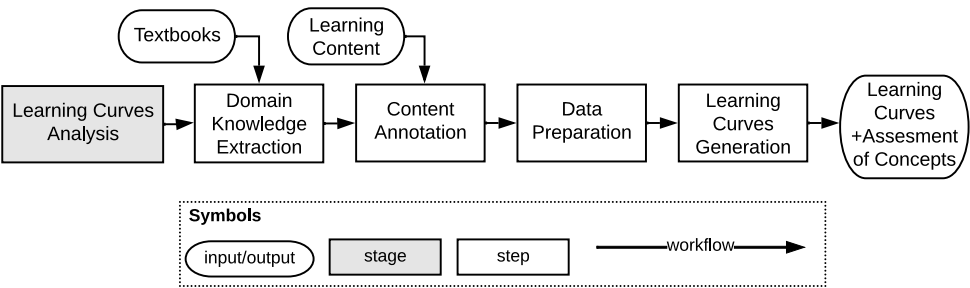


Figure 6.4: Experiment’s procedure for the assessment of concepts as valid KCs (validation phase).

PYTHON model are labeled with their domain specificity to identify relevant concepts (🔗 Chapter 5).

Tables 6.1 and 6.2 present the obtained domain knowledge from the three textbooks. In Table 6.1, *index terms* are the total number of index entries extracted from the back-of-the-book index. *Unique index terms* are the number of index entries after merging duplicates. *Linked entities* is the number of index entries with a linked entity in DBpedia. Finally, *unique linked entities* is the total number of different entities linked to index terms (sometimes multiple terms can be linked to the same DBpedia entity). Table 6.2 shows the number of *concepts* according to their domain specificity.

Table 6.1: Index terms and DBpedia entities identified in the knowledge models.

Model / Data	Index Terms	Unique Index Terms	Linked Entities	Unique Linked Entities
BOOK#1	877	841	300	259
BOOK#2	1063	817	412	357
BOOK#3	566	451	245	210
PYTHON	-	1535	600*	600*

*In the PYTHON model, terms are merged based on the linked DBpedia entities; therefore, the number of linked entities and unique liked entities is the same.

Table 6.2: Identified concepts in the combined knowledge model.

Model / Concepts	in-domain+	related-domain	out-of-domain
PYTHON	266	186	148

From the generated domain knowledge, the *in-domain+* (🔗 Chapter 5, Section 5.3) concepts are selected to annotate the learning activities. Since the *in-domain+*

concepts represent the most important abstractions related to Python programming, this set is a cohesive representation of the domain. Additionally, this set contains different levels of granularity. For example, the set contains the "string" and "while loop" concepts, which represent two very concrete abstractions in Python: a data type and a specific control flow statement. On the other hand, the "expression" concept is more general and represents multiple combinations of values, variables, and operators in Python (e.g., the "boolean expression" and "generator expression" are sub-types of "expression").

Content Annotation

In this step, the learning activities are annotated with the selected concepts to indicate their expected learning outcomes. The annotation is done in a series of rounds by experts (Master-level students, PhD students, and senior researchers) with ample experience in Python programming and teaching. In the first round, an initial annotation of the activities is done. The annotations are reviewed, corrected, and improved in subsequent rounds until a consistent and accurate state is reached. In every round, the prerequisite/outcome relations between concepts according to the order of topics/activities in MG are considered when indicating the concepts to be trained by that activity. Additionally, annotators look at the topic and the expected output of the activities to annotate the relevant concepts.

The final annotation reflects the expected learning outcomes (KCs) for each activity. In total, 54 concepts (out of 266 possible) are used in the annotated activities. This is the final group of concepts that are evaluated in this experiment using learning curves. Table 6.3 lists the concepts grouped by the topic of the activities in which they appeared. Cross-topic concepts are in *italics*. Concepts appearing in only one topic are considered to be fine-grained (e.g., "variable" in "Variables and Operations"). In contrast, concepts used in several topics are considered coarse-grained (e.g., "iteration" in "While Loops" and "For Loops").

Data Preparation

As mentioned before, the data come from students using the MG system in real and diverse settings with no control over the environment. The interactions are not gathered in a controlled experiment in a laboratory. However, they come from students using the system freely with many possible variables (e.g., distractions or the order in which activities were done) and motivations (e.g., honest learning or playing the system to get extra credits). Exercises are also very noisy instruments; errors can happen for multiple reasons. Therefore, the data have to be treated with caution. Reliable learning sequences that help to evaluate the concepts need to be extracted from the data. Sequences with no evidence of learning have to be regarded as noise. It is possible to lose some data using this approach, but eliminating the noise is more critical. The data preparation section describes which techniques are applied to generate reliable learning curves from the noisy data.

First, the interactions in the dataset are grouped into sequences containing all student attempts per concept (based on the procedure used by Sosnovsky & Brusilovsky [252]). All attempts of a student on the same concept are grouped, in the order the answers are given, into a single sequence. For example, if a student gives ten answers to activities training the "for loop" concept, a

Table 6.3: List of concepts evaluated in this experiment, grouped by the topic in which they are used. Topics are shown in the order they are supposed to be trained. Cross-topic concepts are in *italics*.

Topic	Concepts
Variables and Operations	"exponent", "float", "function call", "hello, world", "integer", "operand", "operator", "operator precedence", "string", "value", "variable"
Boolean Expressions	"and operator", "boolean", "boolean expression", "logical operator", "relational operator"
If-Else	"conditional statement", " <i>indentation</i> ", "modulus operator"
While Loops	" <i>indentation</i> ", "iteration", "while loop"
For Loops	"for loop", " <i>indentation</i> ", "iteration"
Nested Loops	" <i>indentation</i> ", "iteration", "nested loop"
Functions	"function", "function parameter", " <i>indentation</i> ", "recursion", "return statement"
Lists	" <i>data structure</i> ", " <i>index</i> ", " <i>list</i> ", " <i>mutability</i> "
Two-Dimensional Lists	" <i>data structure</i> ", " <i>index</i> ", " <i>list</i> ", "list comprehension"
Dictionary	"counter", " <i>data structure</i> ", "key-value pair", " <i>mutability</i> ", "tuple"
Strings	"format string", "string concatenation", "string method", "string operation", "substring"
Values References	"aliasing", "immutability", " <i>mutability</i> ", "reference"
File Handling	"file"
Exceptions	"exception"
Classes/Objects	"class", "constructor", "encapsulation", "inheritance", "instance", "method", "object-oriented", "optional parameter", "self"

sequence for that student and concept is created with the ten answers, regardless of whether the ten answers come from the same activity or multiple ones. In total, 10946 student-concept-attempts are generated. Table 6.4 presents an excerpt of the sequences. Correct answers (outcomes) are marked with 1's, incorrect ones with 0's.

One typical way to interact with Parson's problems is by rapidly changing line by line to guess the correct solution. Highlighted fragments indicate where changes are necessary. This trial-and-error strategy works exceptionally well to get instant feedback. The strategy is good if the students need help understanding a particular

Table 6.4: Student-concept-attempts sequences aggregated from student’s interactions in MG.

Student Id	Concept	Attempt#1	Attempt#2	...
1	"aliasing"	0	1	...
1	"boolean"	0	0	...
...
3	"integer"	1	1	...
...

part of the problem. Little effort and time are required, but it is possible to make a lot of (small) incorrect attempts while reaching the correct solution. Therefore, smoothing is applied to reduce the number of incorrect attempts in the sequences of such activities. Several techniques (e.g., based on the sequence length or the duration of each attempt) achieve similar results, and the simplest is used. Specifically, continuous incorrect outcomes are shortened using the natural logarithm. For example, a sequence of 13 incorrect attempts plus one final correct attempt is changed to three incorrect attempts plus one correct attempt.

Then, noise in the student-concept-attempts sequences is filtered out. There are multiple ways to filter the data. Interactions corresponding to too difficult or easy activities could be removed from the dataset. Also, students who do not show any learning could be withdrawn. Another possible filter is to identify concrete sessions with no learning evidence. Other options include filtering based on the time spent on attempts or behavior of the students across sessions. After trying different filters, only the sequences with evidence of the student’s learning of the concepts are used. For that, each sequence is labeled using four tags: *known*, *understood_strong*, *understood_weak*, and *not_understood*. The first tag identifies students where all interactions with a particular concept are correct—students already know the concept. The second tag identifies students that interact with a concept multiple times and finish with at least two correct attempts in a row—students practice until they show a certain level of mastery. The third tag is used when students interact with a concept multiple times, but they finish with only one correct answer in a row—students practice until getting a correct answer. The final tag identifies students whose answers are all incorrect—students stop before showing any learning. Table 6.5 shows the patterns used to label the sequences.

The *known* and *not_understood* sets of sequences are filtered out from the data. In both sets, there is no learning happening because the concepts are already mastered or not understood completely. On the contrary, *understood_strong* and *understood_weak* sequences contain evidence of learning since the students only stop after answering correctly in multiple attempts. After this filtering, 8079 student-concept-attempts are left (73.8% of all sequences).

After filtering, new attempts are generated by augmenting (i.e., extrapolating) the learning evidence in the student-concept-attempts sequences. Since learning curves aggregate data across multiple students, the support for each attempt tends

Table 6.5: Tagging of student-concept-attempts sequences. Sequences are matched in order starting with *known* and ending with *understood_weak*.

Tag	Pattern (regex-based)	Description
<i>known</i>	1+	All outcomes are CORRECT
<i>not_understood</i>	0+	All outcomes are INCORRECT
<i>understood_strong</i>	$(0^* \rightarrow 1)^+ \rightarrow 1$	The sequence ends with at least two CORRECT outcomes in a row
<i>understood_weak</i>	[10]+	The sequence is a combination of CORRECT and INCORRECT outcomes

to decrease as the number of attempts (i.e., the x-axis) increases [182]. In order to preserve the gathered learning evidence of students who stop practicing a concept earlier than other students, new attempts are introduced in the sequences to have the same number of concept-attempts for each student. A 1 (correct) is inserted in each *understood_strong* sequence until the maximum number of attempts for that concept is reached. For *understood_weak* sequences, the average of the original attempts (the known probability of getting a correct attempt) is inserted. Table 6.6 shows an example of the augmentation of the student-concept-attempts sequences.

Table 6.6: Example of the augmentation of student-concept-attempts sequences.

Tag / Attempts	#1	#2	#3	#4 (augmented)	#5 (augmented)	...
<i>understood_strong</i>	0	1	1	1	1	...
<i>understood_weak</i>	0	0	1	0.333	0.333	...

Learning Curves Generation

In this last step, the learning curves are generated using the processed student-concept-attempts sequences. First, for each concept, their corresponding sequences are selected. Then, the concept error rates at each attempt are calculated using Formula 6.2.

$$\text{error rate } nth \text{ attempt} = 1 - \frac{\text{sum of all outcomes at } nth \text{ attempt}}{\text{total number of outcomes at } nth \text{ attempt}} \quad (6.2)$$

Before plotting the final learning curves, a cut-off point is selected. Given the mentioned decrease in attempt support, a cut-off point is traditionally selected to crop the learning curves before they deteriorate markedly [182]. After generating

and analyzing the raw learning curves, the selected cut-off point is when the number of attempts is less than 25% of the first attempts. This threshold maintains a good balance between the number of attempts and the fit of the learning curves. With a higher reduction, the number of attempts used as evidence decreases. By combining the cut-off point with the augmentation of learning evidence, the learning curves are cropped using the original attempt support (without augmentation) before they plateau due to the constant learning evidence from most students at the latest attempts.

Figure 6.5 presents one of the generated learning curves. There are multiple elements in each plot. The learning curve of the concept is represented with a dashed-blue line. The support (number of students) at each attempt is shown with a dotted-green line. The power law approximation of the learning curve is displayed with a solid-red line. The power law constant (B), slope (α), and fit (R^2) are shown in a gray box. Finally, there are two scales. The one on the left is used for the error rates and the scale on the right for the number of students at each attempt (support).

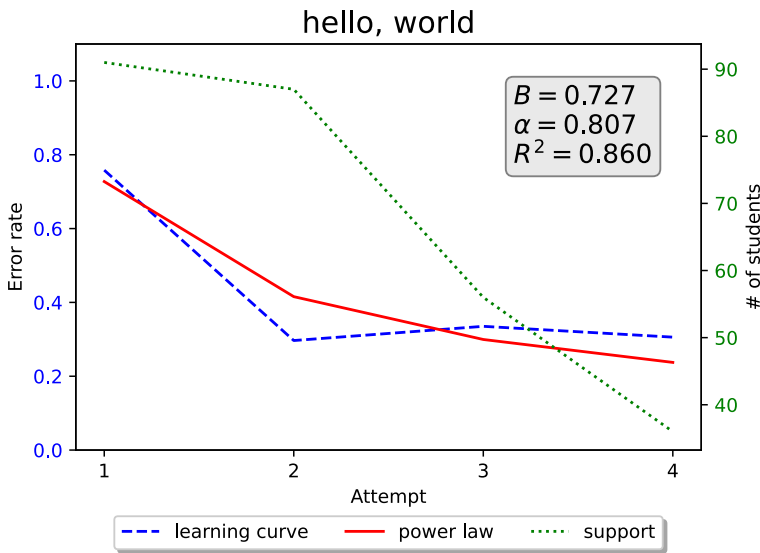


Figure 6.5: Learning curve for the "hello, world" concept.

The next section presents an overview of obtained learning curves and discusses the major findings.

6.4 Results and Analysis

In total, 46 unique learning curves were generated. Eight concepts have the same learning curve as other concepts and were omitted. This situation happened due to closely related concepts which always appeared together in the annotated activities.

Those activities were annotated with too fine-grained concepts, and only a coarser-grained concept is being learned. Section 6.4.2 elaborates on this situation. The omitted concepts are (the related concept is shown in brackets): "boolean" ("boolean expression"), "relational operator" ("boolean expression"), "and operator" ("logical operator"), "operand" ("operator"), "encapsulation" ("object-oriented"), "self" ("object-oriented"), "class" ("object-oriented"), and "method" ("object-oriented"). In the rest of this section, representative learning curves are shown to guide the discussion of the results. However, all learning curves are available in Appendix A.

6.4.1 Cognitive Validity

From the 46 concepts, two of them show a negative slope (α) with a slightly upward trend, meaning there is no learning happening. The 44 remaining learning curves show a positive slope (α) with a downward trend, meaning that for each concept, there is evidence that learning is taking place. The mean fit (R^2) for all 44 positive curves is 0.72 ($SD = 0.24$), which indicates that, in general, **the assessed textbook concepts are cognitively valid units of knowledge**. The obtained fits align with other results reported when dealing with small knowledge units [182, 201, 252]. Figure 6.6 displays a histogram with the positive learning curves' distribution according to their fit.

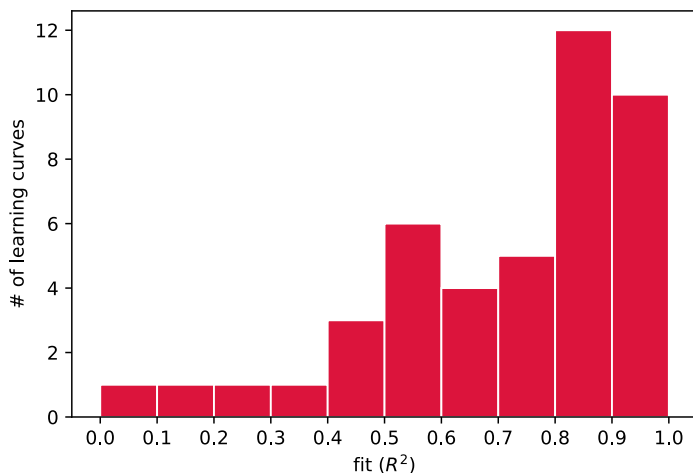


Figure 6.6: Distribution of concepts with downward trend by their fit.

A deeper analysis can be done by classifying the learning curves. Rivers et al. [231] use five categories: *little-data* (not enough data), *already-learned* (start and end with low error rates), *no-learning* (start and end with high error rates and no downward trend), *still-learning* (start and end with high error rates with a downward trend), and *good-learning* (start high but end with low error rates). With this classification, there are two learning curves classified as *no-learning* ("exception" and

"iteration"). The remaining curves are classified as *still-learning*. When there is no filtering of the student-concept-attempts sequences, the *know* and *not_understood* sequences (the ones without any learning) produce lower-quality learning curves. Additionally, these sequences produce zero *already-learned* curves and four *good-learning curves* ("function call", "hello, world", "integer", and "string"), thanks to the increase of correct attempts from the *know* sequences.

However, the classification by Rivers et al. does not describe the fit and slope of the learning curves. A high fit and slope indicate better and more learning. Using these power law parameters, the obtained downward learning curves are classified into *high-quality*, *medium-quality*, and *low-quality*. Learning curves with a high fit (≥ 0.7) and slope (≥ 0.5) are classified as *high-quality*. Learning curves with a low fit (< 0.5) and slope (< 0.4) are classified as *low-quality*. The remaining of the learning curves have a fit higher than 0.5 with low or medium slope and are classified as *medium-quality*. These learning curves are annotated with a downward triangle ($\nabla\alpha$) if they have a low slope (< 0.4). The classification is as follows:

- *High-quality*: "counter", "recursion", "function call", "string concatenation", "hello, world", "string", "tuple", "optional parameter".
- *Medium-quality*: "logical operator", "for loop" $\nabla\alpha$, "modulus operator" $\nabla\alpha$, "aliasing" $\nabla\alpha$, "file" $\nabla\alpha$, "boolean expression" $\nabla\alpha$, "substring" $\nabla\alpha$, "indentation" $\nabla\alpha$, "immutability", "string operation" $\nabla\alpha$, "integer" $\nabla\alpha$, "inheritance", "mutability" $\nabla\alpha$, "index" $\nabla\alpha$, "exponent", "function parameter" $\nabla\alpha$, "return statement" $\nabla\alpha$, "operator precedence" $\nabla\alpha$, "function" $\nabla\alpha$, "list comprehension" $\nabla\alpha$, "instance" $\nabla\alpha$, "operator" $\nabla\alpha$, "list" $\nabla\alpha$, "constructor" $\nabla\alpha$, "float" $\nabla\alpha$, "object-oriented" $\nabla\alpha$, "string method" $\nabla\alpha$, "format string" $\nabla\alpha$, "nested loop" $\nabla\alpha$.
- *Low-quality*: "conditional statement", "variable", "value", "key-value pair", "while loop", "reference", "data structure".

The used classifications provide valuable insights. All downward learning curves (44) are still being learned, and extra practice is needed. From these concepts, eight are *high-quality*, seven are *low-quality*, and 29 are *medium-quality*.

One of the *high-quality* concepts is "function call" ($R^2 = 0.97$, $\alpha = 0.59$). Its learning curve is shown in Figure 6.7. This concept is relatively easy to learn since the initial error rate is 0.63, which after five more attempts, decreases to 0.23. Like Rivers et al. [231], the evidence shows that students learned how to use any function call and are not learning different concepts for every new function. The learning curves from all *high-quality* concepts are steeper than those in the other two categories, showing more and faster learning.

The "for loop" ($R^2 = 0.99$) concept belongs to the *medium-quality* category. Figure 6.8 shows its learning curve. The plot shows that the learning curve smoothly follows the power law. Two main causes can explain the observed learning effect. First, "for loop" is a fine-grained concept representing a very concise programming statement in Python. Second, it has its own topic with learning activities designed to train the concept exclusively. Similarly, other concepts with a very high and smooth fit are "aliasing", "counter", "logical operator", and "recursion".

As mentioned before, the power law's slope indicates the performance improve-

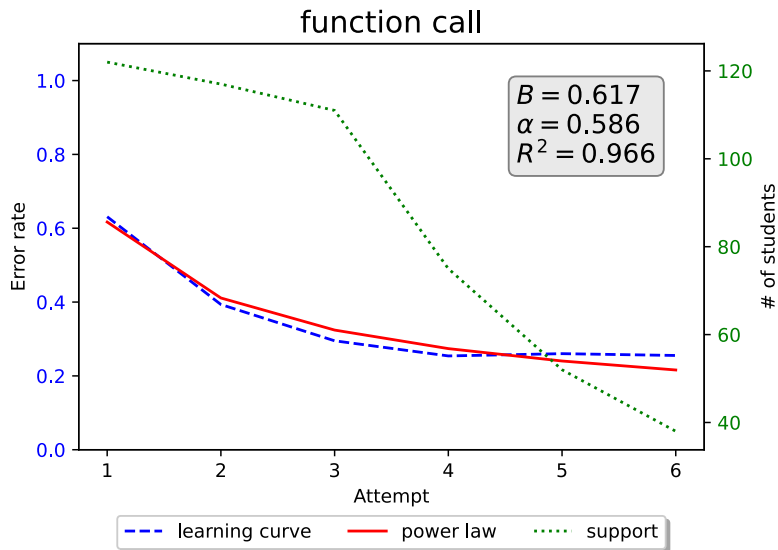


Figure 6.7: Learning curve for the "function call" concept.

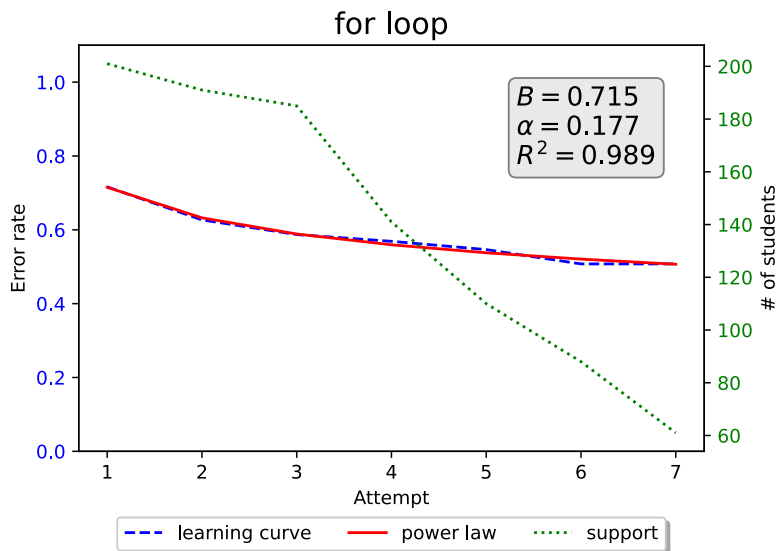


Figure 6.8: Learning curve for the "for loop" concept.

ment speed. For example, for the "for loop" (Figure 6.8), the probability of misapplying the concept clearly decreases with the number of attempts; however, the speed of learning is low ($\alpha = 0.18$). This situation could be related to the difficulty or order of the activities. On the contrary, the learning curve for the "recursion"

concept has both a high fit ($R^2 = 0.97$) and a high slope ($\alpha = 0.75$). The high fit shows that the learning follows the power law, and the high slope shows that the performance improves quickly. Figure 6.9 shows an initial error rate of 0.98 for "recursion", which, at the third attempt, has already decreased to 0.47. Performance speed shown in learning curve analysis is an important aspect to be considered by the content creators to improve the learning systems.

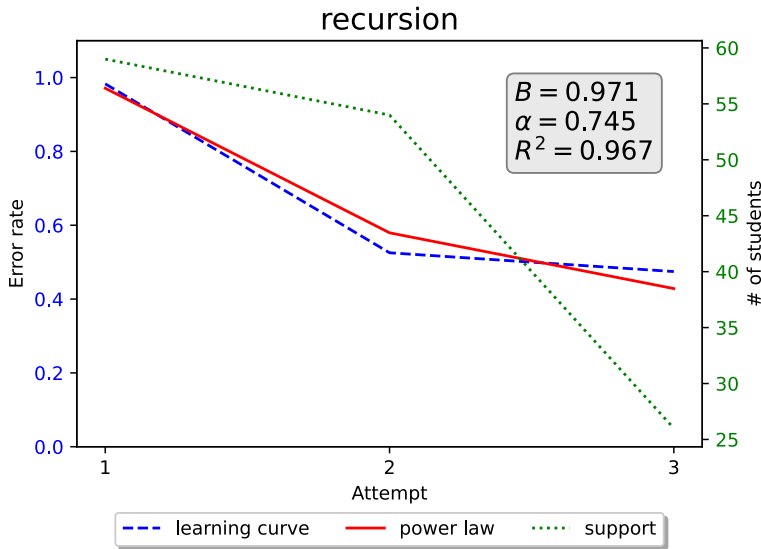


Figure 6.9: Learning curve for the "recursion" concept.

One of the *low-quality* learning curves corresponds to the "while loop" concept. Figure 6.10 shows its learning curve. The fit ($R^2 = 0.41$) shows that learning is taking place, but the low slope ($\alpha = 0.08$) shows that it is happening very slowly. This behavior could mean that students are having trouble understanding while loops and the activities could be improved to enable better learning. The "key-value pair" and "reference" concepts have similar learning curves.

One of the two upward learning curves (*no-learning*) corresponds to the "exception" concept. Figure 6.11 presents its learning curve. The "exception" concept is used in activities that train exception handling in Python and belong to the "exceptions" topic. The learning curve shows a consistent error rate between 0.6 and 0.8. One possible explanation for the learning curve not showing evidence of learning is that the activities are too complex and diverse, involving a lot of other different concepts, which can be a source of errors. In the activities, exception handling is mixed with prerequisite concepts ("for loop", "list", and "function") that are still being learned (they belong to the *still-learning* category).

Additionally, the good matching between textbook concepts and external learning activities shows that the conceptual representation extracted from textbooks in a specific domain has good coverage and focus. From the set of 266 *in-domain* + concepts, it was possible to annotate all 85 learning activities with their expected

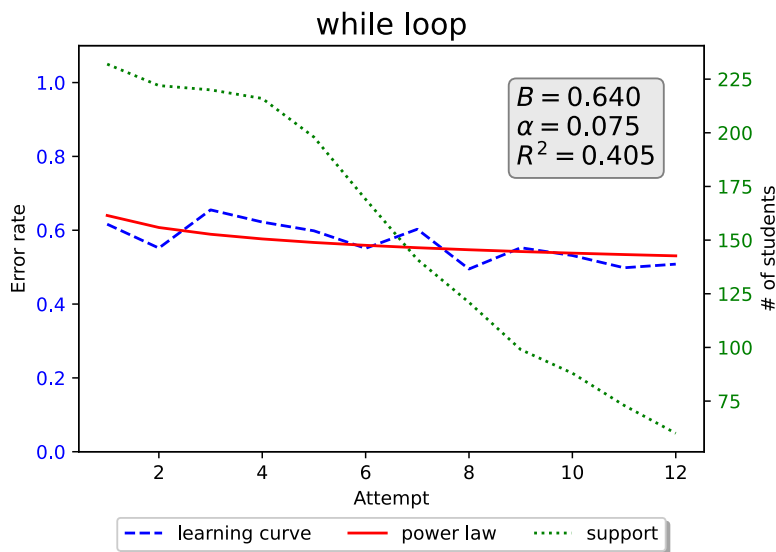


Figure 6.10: Learning curve for the "while loop" concept.

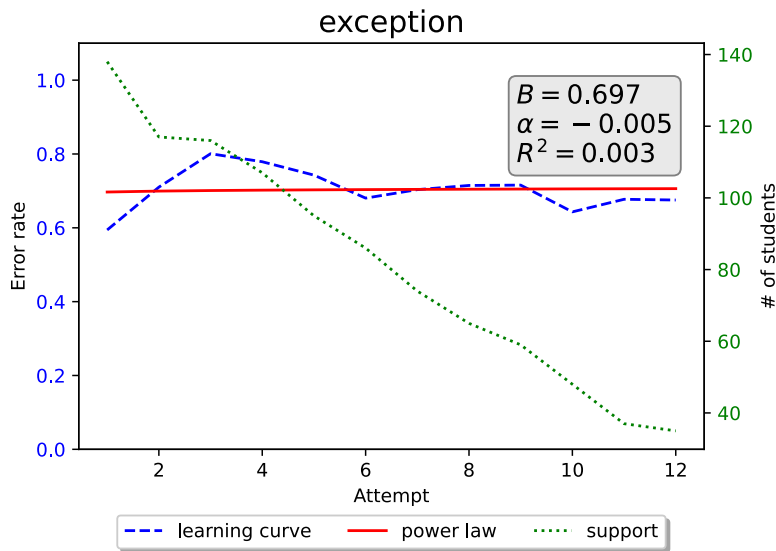


Figure 6.11: Learning curve for the "exception" concept.

learning outcomes. From the 54 concepts used in the final annotation, all but two produced acceptable learning curves. This set of concepts represents valid KCs for knowledge assessment, and they can be used anywhere domain-specific knowledge is required. As a note, it is crucial to emphasize that the textbooks used in this

experiment were not selected to match the learning activities in MG beforehand.

In summary, the classification and analysis of the learning curves showed that concepts extracted from textbooks model the learning taking place by students while solving activities.

6.4.2 Granularity

From the granularity perspective, all single-topic concepts (Table 6.3) can be seen as fine-grained and have acceptable learning curves (other than "exception"). The more coarse-grained concepts are a particular case and are discussed in this subsection.

Coarse-grained concepts (shown in *italics* in Table 6.3) are used in multiple topics. Out of these concepts, all but "iteration" produce downward learning curves. However, their learning curves are not smooth. Figure 6.12 shows the learning curve for the "indentation" concept. In Python, indentation is semantically meaningful and is used to indicate a block of code in many statements or expressions. The "indentation" concept is linked to the "conditional statement", "while loop", "for loop", and "function" concepts. As seen in Figure 6.12, the learning curve is uneven with multiple upticks (bumps). The other coarse-grained concepts show similar learning curves.

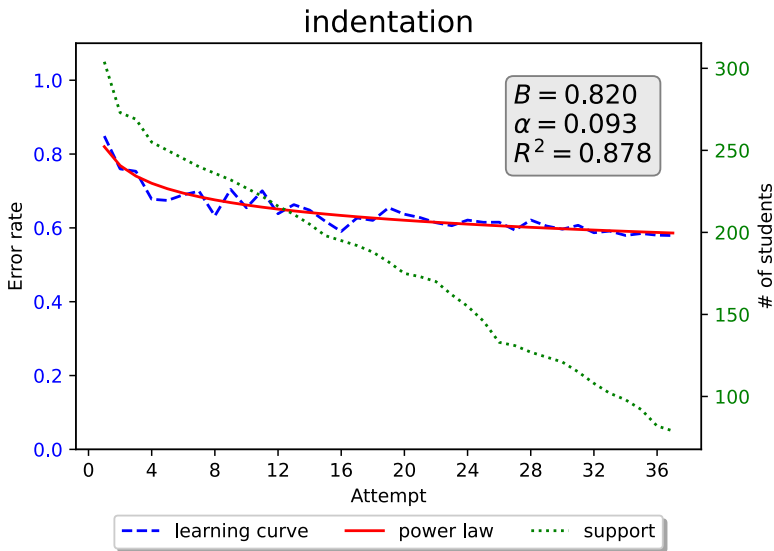


Figure 6.12: Learning curve for the "indentation" concept.

The bumpy learning curves could be explained by the fact that the concepts are being learned within a context of associated concepts. Once a student has learned one of the associated concepts (e.g., "conditional statement" in the case of "indentation"), they move to a different concept (e.g., "while loop"), and the learning process starts again for this associated concept. In other words, the probability of making a mistake is high every time a new associated concept is introduced. After all the as-

sociated concepts have been trained, the central concept comes to a close, showing a downward trend. The obtained learning curve for these coarse-grained concepts agrees with Martin et al. [182] and Sosnovsky & Brusilovsky [252], which observed worse learning curves for more general groupings.

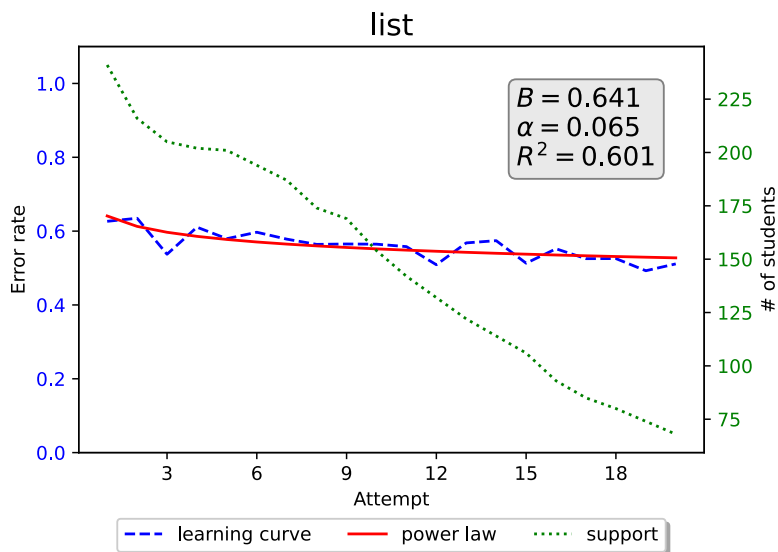


Figure 6.13: Learning curve for the "list" concept.

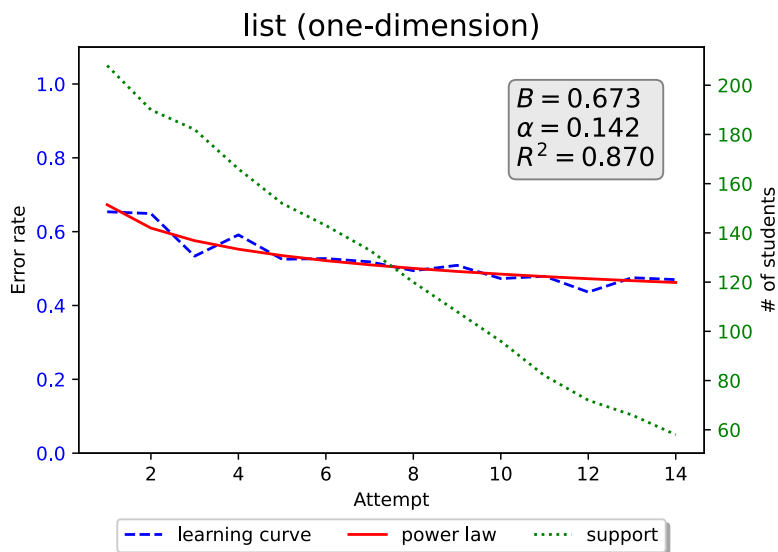


Figure 6.14: Learning curve for the "list (one-dimension)" concept.

The "list" and "index" concepts are an interesting case. At first sight, these concepts

are fine-grained. However, they were used for both one- and two-dimensional lists. The learning curve for the "list" concept is shown in Figure 6.13. The learning curve behaves as in the "indentation" concept. Even though two-dimensional lists are more complex than one-dimensional, they were annotated with the same concept since there are no particular concepts for describing multi-dimensional lists in the used textbooks. At least within the MG system, one alternative is to combine concepts with topics to have more fine-grain annotations. For example, the "list" concept can be separated into "list (one-dimension)" and "list (two-dimensions)". The effect of this specialization of concepts can be seen in Figure 6.14. The learning curve for the "list (one-dimension)" concept was computed using only the activities belonging to the "lists" topic. As observed, this learning curve of just one-dimensional lists is smoother and has a higher fit ($R^2 = 0.87$) than its coarse-grained version ($R^2 = 0.60$). Tacoma et al. [257] observed similar results when KCs with upward learning curves were separated into smaller units, resulting in several downward learning curves.

As mentioned before, eight concepts produced identical learning curves to other concepts. After analyzing the learning curves and the annotation, it is possible to conclude that some activities were annotated with highly fine-grained concepts. For those activities, learning is better modeled as an aggregation of several concepts. For example, activities in the "Variables and Operations" topics were annotated with the "operand" and "operator" concepts. These two concepts always appear together, and it is only possible to have one with the other in the activities. In this case, a coarser-grained concept, like "expression", would have been better. In another example, four concepts that describe object-oriented features produced the same learning curves as the "object-oriented" concept. This last concept groups all the abstractions related to object-oriented programming and is enough to describe them.

The granularity of the entities in DBpedia can also be examined thanks to the generated learning curves. One of the worst learning curve corresponds to the "conditional statement" (if/else) concept. Its learning curve is shown in Figure 6.15. This learning curve shows poor quality in terms of both slope and fit ($R^2 = 0.03$, $\alpha = 0.02$), which probably means the concept does not correspond to what is being learned [182]. The activities annotated with this concept include not only simple *if statements*, but also *elif*, *else*, and *nested conditionals*, which are more complex structures. Even though the *PYTHON* model includes different terms for the "conditional statement", "if statement", "elif keyword", and "nested conditional" abstractions, the four of them are linked to the same entity in DBpedia—the conditional (*Conditional_ (computer_programming)*). Therefore, all the mentioned abstractions are grouped under the "conditional statement" concept. The poor learning curve suggests that the "conditional statement" concept is too broad, and more fine-grained concepts should be used for activities in the "if-else" topic. This situation also hints that textbooks contain fine-grain concepts, but the external model–DBpedia, uses more coarse-grained concepts.

It has been shown that the concepts extracted from the textbooks have different granularities. Textbooks provide both fine-grained and coarse-grained concepts. Both types are valid units of knowledge, but in general, fine-grained concepts model knowledge better.

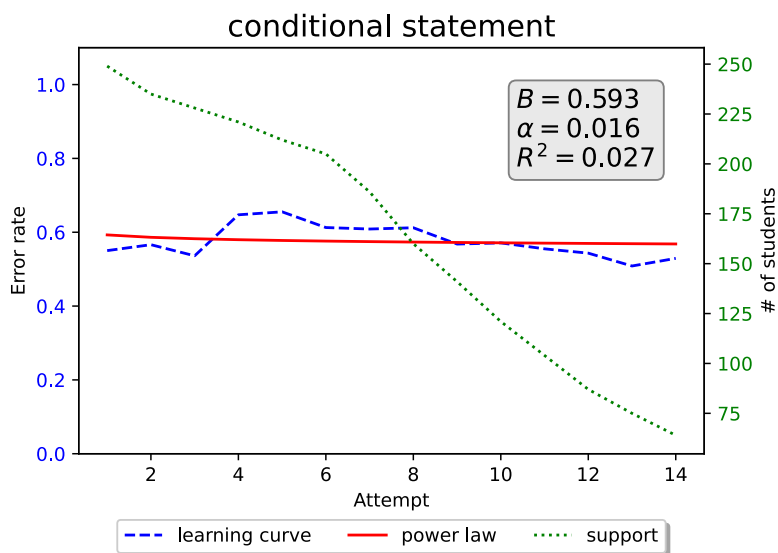


Figure 6.15: Learning curve for the "conditional statement" concept.

6.4.3 Coverage

The coverage of the textbooks as source of concepts has also been revisited in the experiment described in this chapter (☞ Chapter 4, Section 4.4.3 first discussed this property). Figure 6.16 shows a stack plot with the percentage and number of concepts used for the final annotation of activities in each textbook. *BOOK#2* contains 94% of the concepts, while *BOOK#1* only contains 54%. Table 6.7 presents more information on the provenance of the concepts. Five concepts solely appear in one of the three textbooks: one only in *BOOK#1*, three in *BOOK#2*, and one in *BOOK#3*. Twenty-three concepts appear in two textbooks, and 26 are shared by all three. Since textbooks in the same domain may have a different focus, the selection of textbooks can seriously impact the final list of concepts available for annotation or domain modeling. For example, only using *BOOK#1* as the source of concepts will result in an incomplete and insufficient representation of the domain. Using only *BOOK#2* will contribute to most concepts, but not all. *BOOK#3* deliberately omits the "recursion" topic/concept, which can be found in the other two textbooks.

Aggregating the knowledge from multiple textbooks increases the coverage of the domain, resulting in a more accurate and broad set of concepts.

6.4.4 Diversity of Concepts

A direct consequence of good domain coverage is a diversity of concepts. In the context of Python, the extracted concepts from the textbooks can be grouped into two categories: syntactic and semantic. The former group corresponds to concepts directly related to the program constructs available in the Python language. For example, "while loop", "variable", "modulus operator" and "integer". The latter refers

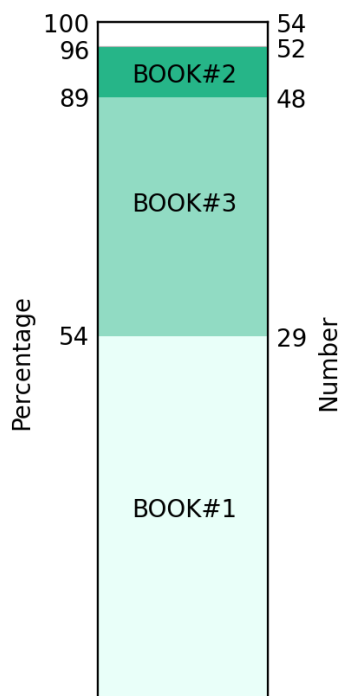


Figure 6.16: Coverage of the textbooks as source of concepts.

Table 6.7: The provenance of the used concepts.

Number of Books	Number of Concepts	Source		
		BOOK#1	BOOK#2	BOOK#3
1	5	1	3	1
2	23	2	23	21
3	26	26	26	26

to concepts that identify higher-level abstractions. For example, "object-oriented", "search algorithms", "refactoring", and "xml". Concepts from both categories can be used to annotate not only the building blocks needed to create algorithms and programs but also their meaning and purpose.

In this experiment, the used learning activities were annotated originally in MG with concepts derived automatically from the Python parser. However, the parser is only concerned with the syntax of the programs. This situation limits the scope of the annotation. For example, one activity from the "Classes/Objects" topic about *inheritance* is annotated with the following MG concepts: "ClassDef" and "FunctionDef". The same activity was annotated for this experiment with the following concepts:

"class", "self", "method", "object", "encapsulation", "object-oriented", "instance", and "inheritance". The original annotation is insufficient because it does not have a concept for inheritance. Additionally, "method" is more accurate than "FunctionDef" since "method" is the correct terminology used in object-oriented programming. Another example is the activity from the "Values References" topic shown in Figure 6.17. In this activity, the "aliasing" concept must be understood to solve the problem correctly. However, the activity is not annotated in MG with any concept indicating the necessary knowledge to solve it. This situation results in a poor representation of the knowledge involved in the activity.

In conclusion, textbooks provide diverse concepts that can be used for precise domain annotation.

```
def main():
    numbers = [1, 3, 5, 7, 11, 13]
    print(numbers)
    original = numbers
    numbers[2] = 2
    print(original)
    original[2+1] = 4
    print(numbers)

main()

What is the output?
```

Figure 6.17: Activity annotated with the "aliasing" concept.

6.5 Conclusion and Future Work

This chapter has explored the validity of concepts extracted from textbooks as units for knowledge assessment. Both the procedure and the results of the experiment were described and discussed. Specifically, a learning curve analysis showed that textbook concepts in the *Python programming* domain measure the students' learning while interacting with different activities (cognitive validity). Additionally, the studied concepts displayed different levels of granularity in the domain. Finally, the experiment also showed that using multiple textbooks as a source of knowledge increases the domain's coverage and diversity of concepts. In conclusion, this chapter provided strong evidence of the richness of the extracted textbook models as a source of concepts for domain modeling and assessment.

Future work includes performing a more complex and controlled experiment with an AIES designed with textbook concepts as the core of the domain and student models. Additionally, the same learning curve analysis could be performed with concepts from different domains to generate more insights into the universal validity of textbook concepts as KCs.

CHAPTER 7

Lights, Camera, Action! Applications of Textbook Knowledge Models

Abstract - There are many potential applications for the textbook knowledge models described in this thesis. This chapter presents three systems that use the generated models to offer different intelligent services. First, Interlingua is a system that creates a language bridge to recommend relevant reading material in the mother tongue of students learning in a foreign language. Second, InTextbooks converts PDF textbooks into intelligent educational Web resources to offer interactive and adaptive content. Third, the integration between InTextbooks and a Python programming practice system creates connections between the sections of the textbooks and smart interactive activities. The analyzed systems show that the extracted knowledge models can support the development of AIES in multiple ways.

This chapter is based on the following publications:

Alpizar-Chacon, I. & Sosnovsky, S., “Interlingua: linking textbooks across different languages”, in: *Proceedings of the First Workshop on Intelligent Textbooks*, vol. 2384, CEUR-WS, 2019, pp. 104–117.

Alpizar-Chacon, I., Hart, M. van der, Wiersma, Z. S., Theunissen, L. S. & Sosnovsky, S., “Transformation of pdf textbooks into intelligent educational resources”, in: *Proceedings of the Second Workshop on Intelligent Textbooks*, vol. 2674, CEUR-WS, 2020, pp. 4–16.

Alpizar-Chacon, I., Barria-Pineda, J., Akhuseyinoglu, K., Sosnovsky, S. & Brusilovsky, P., “Integrating textbooks with smart interactive content for learning programming”, in: *Proceedings of the Third Workshop on Intelligent Textbooks*, vol. 2895, CEUR WS, 2021, pp. 4–18.

7.1 Introduction

Up to this chapter, this thesis has described the technical details of the different stages for the extraction of knowledge models from textbooks. The high quality (accuracy, semantics, coverage, specificity, cognitive validity, and granularity) of such models has been stated through multiple evaluations. Now, it is time to explore the applicability of the generated models.

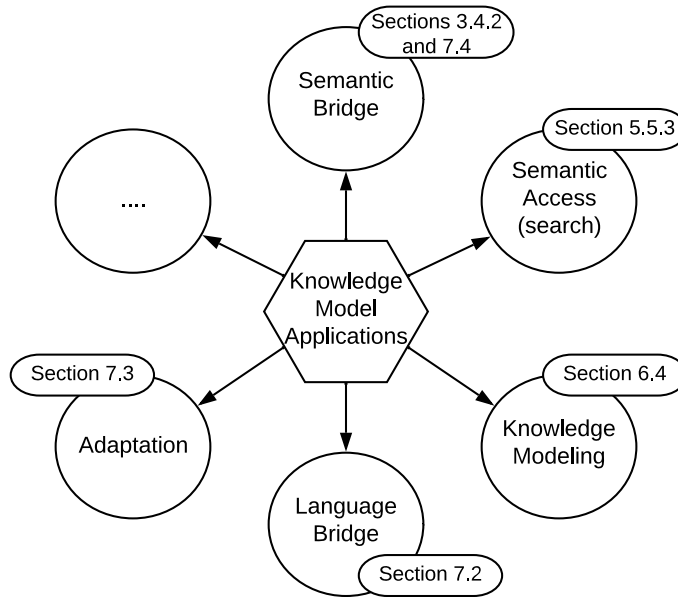


Figure 7.1: Applications of the extracted knowledge models and the sections where they are discussed.

Thanks to the rich domain and semantic knowledge of the models, they have multiple applications. Figure 7.1 shows some of them. First, the models can act as the semantic bridge between different sources or types of content, eliminating the need for time-consuming manual processes. The domain terminology included in the models can be used to link similar sections across textbooks in the same domain (🔗 Chapter 3, Section 3.4.2). Similarly, the model's domain terminology can be the bridge to link different types of content (e.g., textbooks and learning activities) annotated with different glossaries in the same domain (🔗 Section 7.4). Second, the different sections of the textbooks are annotated with domain terms and semantic information (e.g., categories or DBpedia entities) in the models, which facilitates the semantic access of such elements in search (retrieval) tasks (🔗 Chapter 5, Section 5.5.3). Third, since the knowledge models contain the most relevant concepts in the domain, they can be used to construct domain and user models, which are central elements in the design of AIES (🔗 Chapter 6, Section 6.4). In addition, given that the knowledge models can be connected to external multilingual models (e.g., DBpedia or the ISI Glossary), they can act as a language bridge to connect content in different

languages (🔗 Section 7.2). Moreover, the knowledge models' hierarchy of elements (from words to sentences to paragraphs to subchapters to chapters) can be used to support adaptation, where only the relevant components for each student are displayed according to her or his current needs (🔗 Section 7.3). Finally, the knowledge models can be used in any task requiring high-quality domain knowledge.

Furthermore, to delve into the diverse applications of the knowledge models, this chapter describes three different AIES supported by knowledge models extracted from textbooks. These systems are designed as a proof of concept to show that it is possible to build applications with the extracted knowledge models. The systems and the role of the knowledge models in each one are summarized as follows.

Interlingua. In this AIES, students can study textbooks in a foreign language supported by on-demand access to relevant reading material in their mother tongue. The generated knowledge models support the connections between textbooks in different languages. Each textbook model is linked to a glossary of domain terms in multiple languages. This glossary is used as a reference model where the textbook sections are mapped, regardless of the language. This mapping allows semantic links between each textbook section to the most similar sections in textbooks from different languages.

InTextbooks. This system automates converting PDF-based textbooks into intelligent educational Web resources. InTextbooks uses the elements in the knowledge models to first convert the PDF textbooks to HTML representations, where each element is precisely identified in a fine-grained DOM structure. Then, interaction and adaptation are offered for each element in the models. For example, when clicked, important words (domain terms) can display additional information; or a personalized navigation path is shown thanks to the models' structural information (ToC).

InTextbooks/P⁴. This system is a two-way integration of textbooks with "smart" interactive content using InTextbooks and P⁴—a personalized Python programming practice system. In this case, the generated knowledge model acts as a semantic bridge for automatic linking. Domain terms from the textbooks are linked to concepts in an ontology. This linkage between two different glossaries allows for displaying relevant learning activities while reading chapters of the textbook, while users interacting directly with the learning activities have access to related reading material.

The main contribution of this chapter is the overview of how the extracted textbook knowledge models support different AIES. This chapter is organized as follows. Sections 7.2, 7.3, and 7.4 present the details of each of the three systems. Section 7.5 concludes this chapter.

7.2 Interlingua: Linking Textbooks Across Different Languages

7.2.1 Motivation

Two parallel trends exist in the current European Union (EU) education, independent one from another, originating from different conditions, yet leading to a shared

outcome. From the socio-economic perspective, EU promotes ever-increasing mobility, especially when it comes to younger population. The Bologna process [33], the “Youth on Move” initiative [89], and students exchange programs like Erasmus contribute to the vision of a joint European education ecosystem, where students from all EU countries freely and actively engage in educational programs and individual courses across borders and cultures.

From the pedagogical (and technological) perspective, new forms of learning have emerged supported by information and communication technologies. They facilitate free and easy access to learning materials and promote more central and active role of a learner. Initiatives like Open Educational Resources (OER) and phenomena like MOOCs shape the new educational reality where students have more choice and flexibility in terms of which textbook to read, which course to take, and which skill to acquire. As a result they become less dependent on the actual institution issuing a degree.

These two trends reinforce each other and jointly contribute to a much-desired outcome of more scalable, sustainable, and affordable education. However, they also lead to a potential problematic situation that is occurring more often as more international students enroll in formal university courses and/or free MOOCs taught in a foreign language. Studying a course from an unfamiliar university/program is challenging enough. It might be taught on a new (more abstract or intensive) level. It might require students to have prerequisite knowledge and skills that they have not acquired yet. For foreign students, this transition to new course requirements is aggravated by the necessity to learn material in a foreign language that they did not use when taking the prerequisite courses. A foreign student inevitably faces a certain language barrier amplified by the mismatch in the background knowledge and terminology. Unfortunately, the current tradition of resolving these difficulties is hardly efficient - teachers report that non-native students are allowed to bring dictionaries to regular classes and exams. As a result, the educational system promotes student mobility on the level of policies, but does not sufficiently support it in on the individual level.

An effective remedy to this problem is the provision of international students with multilingual access to instructional material, where educational resources in a language of a course are accompanied by resources in their native language. Two principal ways to achieve this are the translation of the original resource and linking two corresponding “inter-lingual” resources. The translation-based approach will not help solve the problem: manual translation is not scalable, and machine-based translation is not yet capable of producing results of adequate quality in a narrow academic domain. This section presents Interlingua, a solution based on automated semantic linking of related educational resources across languages with the main focus on textbooks. The system extracts the knowledge models from textbooks and then links them to an external reference ontology. The reference model is then used as a multilingual bridge across textbooks in different languages to link similar sections. Probability theory and statistics have been chosen as the target domain. One reason for it is the popularity of this subject in teaching programs for many technical and social science degrees. Another reason is that this subject uses a lot of specific terminologies, both new and borrowed from prerequisite parts of mathematics.

7.2.2 Background

Several projects experimented with semantic linking of relevant textbooks written in the same language. Guerra et al. [117] used a probabilistic topic modeling approach to extract topic models from textbook and use them to link sections and subsections across multiple textbooks. Later, Meng et al. [193] explored different content modeling approaches for textbook linking: a term-based approach (each word is considered as a knowledge component), LDA (latent topics representing knowledge components), and a concept-based approach (author-assigned keywords in scientific publications representing knowledge components). Also, an ensemble of the three approaches was used. The semantic and the combined approaches achieved valuable linking performance.

7.2.3 The Application

Architecturally, Interlingua consists of two large components. The offline component (ingestion) performs the tasks of textbook modeling and linking, while the online component (interface) supports students' interaction with the content of linked textbooks. This subsection describes the details of both components.

Ingestion of Textbooks

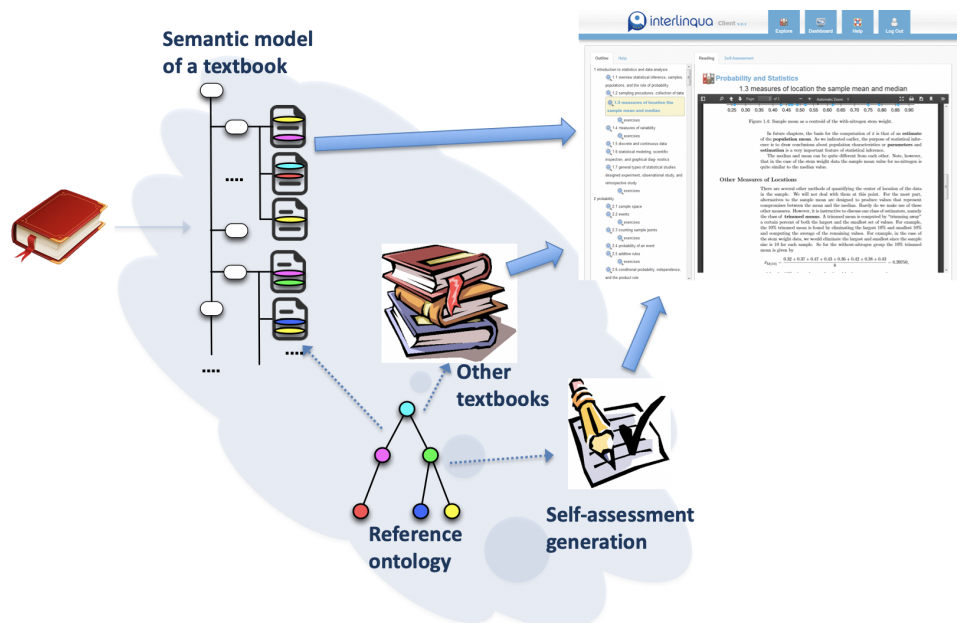


Figure 7.2: Overall process of adding a textbook into Interlingua.

The overall process of adding a new textbook into the Interlingua content repository is depicted by Figure 7.2. The only action performed manually is the upload of textbook files. A teacher decides which textbooks and in which languages should

be available for the students of the course. After a textbook file is submitted, its knowledge model is extracted and enriched with semantic information from DBpedia using the approach presented in this thesis (☛ Chapters 3 and 4). The extracted index terms are used as semantic anchors to link pages and sections of the textbook to the concepts of the reference ontology and through them to other textbooks available in the content repository. Finally, the self-assessment component uses the information from DBpedia to generate multiple choice questions related to the index terms explained by the current section of the textbook. Once processed and stored this way, the textbook becomes accessible through the student interface. Apart from the textbook knowledge models, Interlingua uses a reference ontology to perform the textbook linking.

The ISI Glossary is used as a reference ontology. Initially, each term from the glossary was linked to its corresponding entity in DBpedia. Then, each index term from the knowledge model is compared against each term of the ISI Glossary, and when the cosine similarity is over 90%, both terms are linked. This strategy allows for constructing a VSM over the terms of the ISI Glossary to create the links among the different parts of textbooks. The VSM consists of several documents (each section) per textbook in the rows, and the entries of the ISI Glossary as the columns. The weight for each document's term is calculated from the strength of the association between the term and the document. The VSM is improved by pruning the terms in the glossary that are not found in the corpora. After the VSM is created, it is split into matrices for each language. A similarity matrix is created for each language pair by multiplying the two appropriate parts of the VSM together and then normalizing the resulting matrix. These matrices contain the similarity value for any pair of sections of textbooks from different languages. Finally, each textbook's section is linked to the five most similar documents in each target language to create the links among the textbooks in different languages.

Interlingua also has an assessment engine that generates Multiple Choice Questions (MCQ) for the learners to examine their understanding of the terminologies related to the section that they are currently reading. The basic MCQ asks the learners to select the correct translation of one of the introduced concepts in the current section into their native language. Each question has only one right answer and several distractors. To generate the questions the assessment engine uses the ISI Glossary to get the labels in different languages for the same concept. Distractors are chosen using the relations among concepts obtained from the enrichment of the model to select reasonably difficult distractors. In other words, the semantic distance between the correct answer and a distractor in the model should not be very long and the lengths of both textual labels should be comparable.

Student Interface

When a student accesses the Interlingua client, the entry page shows a list of available textbooks in the target language of study. A student can indicate her language of study and the mother tongue. Currently, Interlingua uses textbooks in English [69, 282], French [142, 277], German [84, 91], Spanish [281], and Dutch [48, 178].

A student selects one of the available textbooks to open the textbook navigation

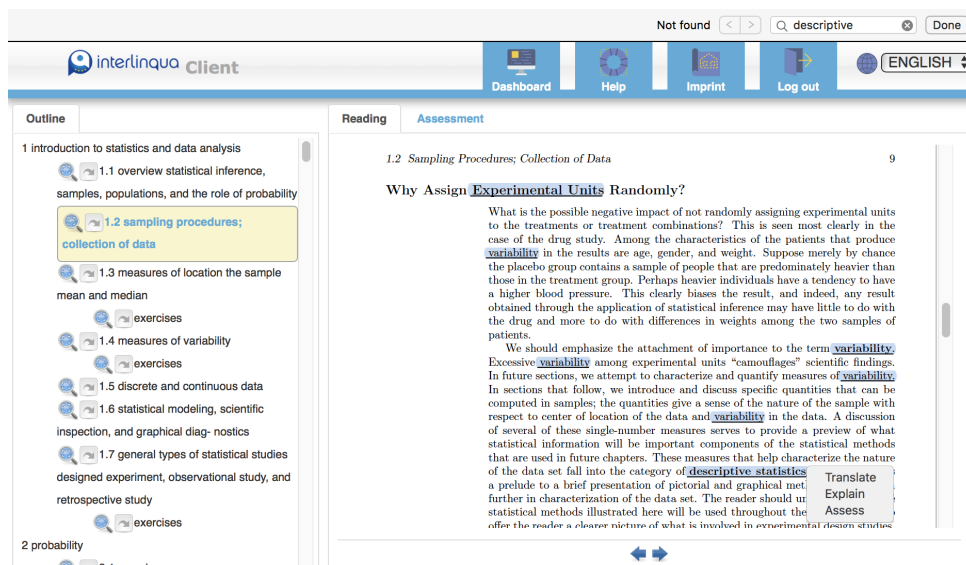


Figure 7.3: Textbook navigation page.



Figure 7.4: Related readings displayed in a pop-up.

page (Figure 7.3). It consists of the outline panel on the left side and the content panel with multiple tabs on the right side of the window.

In the outline panel, a student can browse through sections and subsections of the textbook. She can click on a magnifying glass icon annotating each section title to load the respective section into the content panel. The arrow icon is used to display a list of related readings of the selected subsection in the mother tongue of the student (Figure 7.4).

The student can browse through a subsection's content in a similar way to any standard PDF viewer application. Index terms are recognized and highlighted in the content to indicate that additional interaction with them is possible. When a student clicks on a highlighted word, an action menu appears with three available

options: translate, explain, and assess (Figure 7.5). The *translate* action opens a pop-up window that presents the translation of the term into the mother tongue of the student, and the definition of the term extracted from DBpedia (if this index term has been found in DBpedia). The *explain* action opens a pop-up window that gives access to the sections of the textbooks in the mother tongue of the user in which the term is explained. The *assess* action generates and displays in a pop-up window an question about the translation of the term into the mother tongue of the user.

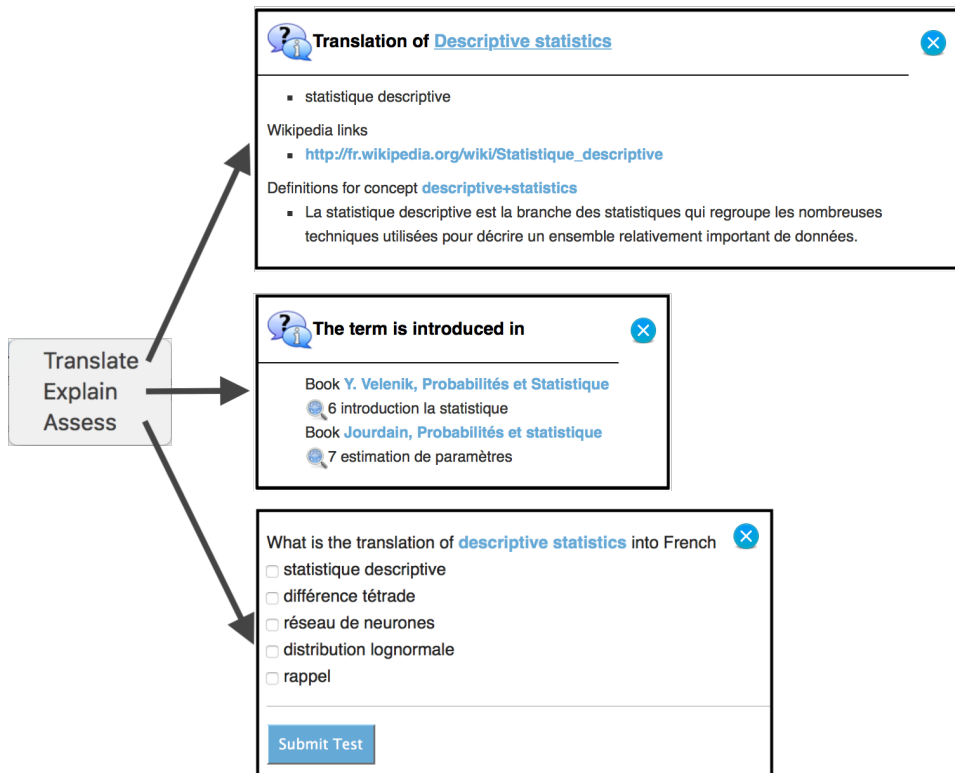


Figure 7.5: Action menu and pop-ups available for highlighted index terms.

As mentioned before, in the navigation page a student can also open related readings for the current section in her mother tongue (Figure 7.3). A pop-up window suggests a list of related readings (Figure 7.4). If a student selects any of the suggested links, a corresponding subsection will be loaded in a new tab. The student can switch between the reading tabs that now contain the related content in two different languages.

Finally, the user can click on the "Assessment" tab in the content panel to generate an assessment composed of several MCQs related to the content of the currently browsed section (the questions are similar to the one at the bottom pop-up in Figure 7.5).

7.2.4 Discussion

Interlingua is a system that provides international students with access to related textbook material in different languages. Interlingua exploits the model's domain knowledge to create a language bridge between all available textbooks regardless of the language. The set of index terms associated with each section implicitly encodes its topic, which is captured using a VSM. Since terms can have multiple synonyms and to account for the multilingual context, the terms are mapped to the ISI Glossary. This external reference model, along with the domain information from the textbook models, allows for the linking of similar sections across textbooks, independently of the language of the textbooks.

From a broader perspective, Interlingua is an example of a service that can be built on top of linked knowledge models extracted from related textbooks. However, the corpus of semantically linked high-quality educational content used in the system can also implement a range of different intelligent services: adaptive navigation, recommendation of textbook content, enrichment of textbook content with external (interactive) educational resources, and extraction of different types of learning objects from the textbook themselves.

Finally, the Interlingua technology could be applied to other domains besides probability theory and statistics.

7.3 InTextbooks: Transforming PDF Textbooks into Interactive Educational Textbooks

7.3.1 Motivation

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 [71]. Some of these textbooks come equipped with additional intelligent services built around them to support enhanced search [251], easier navigation [42], interactive content [88], and, ultimately, better learning [230]. 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.

The research presented in this thesis is an initial step to address this problem. The presented approach extracts automatically machine-readable knowledge models from PDF textbooks. However, while the approach is capable of extracting a semantic model of a textbook, link it to other models and essentially provide a backbone for implementing intelligent services (e.g., adaptation), it lacks an important component that performs the actual conversion of static PDF files into interactive

resources that can be published on the Web.

This section presents InTextbooks (for Intelligent Textbooks)—the system that can perform the complete transformation of PDF textbooks into online intelligent educational resources. After the extraction of the knowledge model from a PDF textbook, InTextbooks converts the model into an HTML/CSS representation with an enriched fine-grained Document Object Model (DOM)¹. Every structure/content/domain element of the textbook (words, lines, fragments, pages, chapters, subchapters, 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 knowledge 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.

7.3.2 Background

Basic conversion from PDF to HTML documents is a well-known task. In 2003, Rahman & Alam [224] 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 et al. [180] presented a system for converting PDF books to the ePub format, which produces XHTML² files. They focused on 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 tools³ and libraries (☛ Section 7.3.3) 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.

A good interaction design is an important factor affecting user acceptance of digital textbooks [58]. 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 [74, 102, 107, 154], to intelligent services such as semantic search [75] and adaptive navigation support [40]. Semantically-enriched textbooks are capable to support meaningful linking and rearrangement of content [112], semantic search and retrieval of relevant learning objects [191] and even targeted inquiry, exploration and comparison of important notions in the domain [56]. Adaptive textbooks trace progress of their readers and maintain composite models of their knowledge to provide personalized content or interaction. For example, Brusilovsky & Eklund [42] presented an online textbook with adaptive navigation support that annotates links to its resources with indicators to inform

¹<https://www.w3.org/DOM/DOMTR>

²<https://www.w3.org/TR/xhtml1/>

³<https://www.pdf2html.org/>, <https://pdf.io/pdf2html/>, etc.

students about the individual educational value of the linked resources. Other examples of adaptation can be found in De Bra [65], Kavcic [146], and Ullrich & Melis [273].

7.3.3 The Application

The Intextbooks system consists of two main groups of components. The offline components (ingestion and conversion) perform the tasks of textbook modeling and conversion to HTML, while the online components (Web-reader) support students' interaction with the textbooks. Figure 7.6 presents the overall architecture of the system.

The ingestion and conversion components take a PDF textbook and, first, extract its knowledge model using the approach presented in this thesis (👉 Chapters 3 and 4). Then, the PDF textbook is converted into an HTML representation. As the last step, the textbook model and the HTML representation are synchronized, meaning all elements of the knowledge model are connected to the DOM elements of the HTML version of the textbook. After that, both the knowledge model 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.

The remainder of this subsection describes the details of both sets of components.

Ingestion and Conversion

Textbook Model Extractor. For each textbook, this component first extracts a knowledge model and then enriches it with semantic information from DBpedia (👉 Chapters 3 and 4). The resulting model is represented as an RDF-enriched TEI document. In the model, each word, line, text fragment, page, (sub)section, index term, ToC entry, etc. is recognized individually. Each element has a unique ID. These IDs will be used to synchronize the model and HTML representations of the textbooks.

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 [52]. Several open libraries have been developed to perform the low-level processing of these PDF primitives and converting them into an HTML representation. Four li-

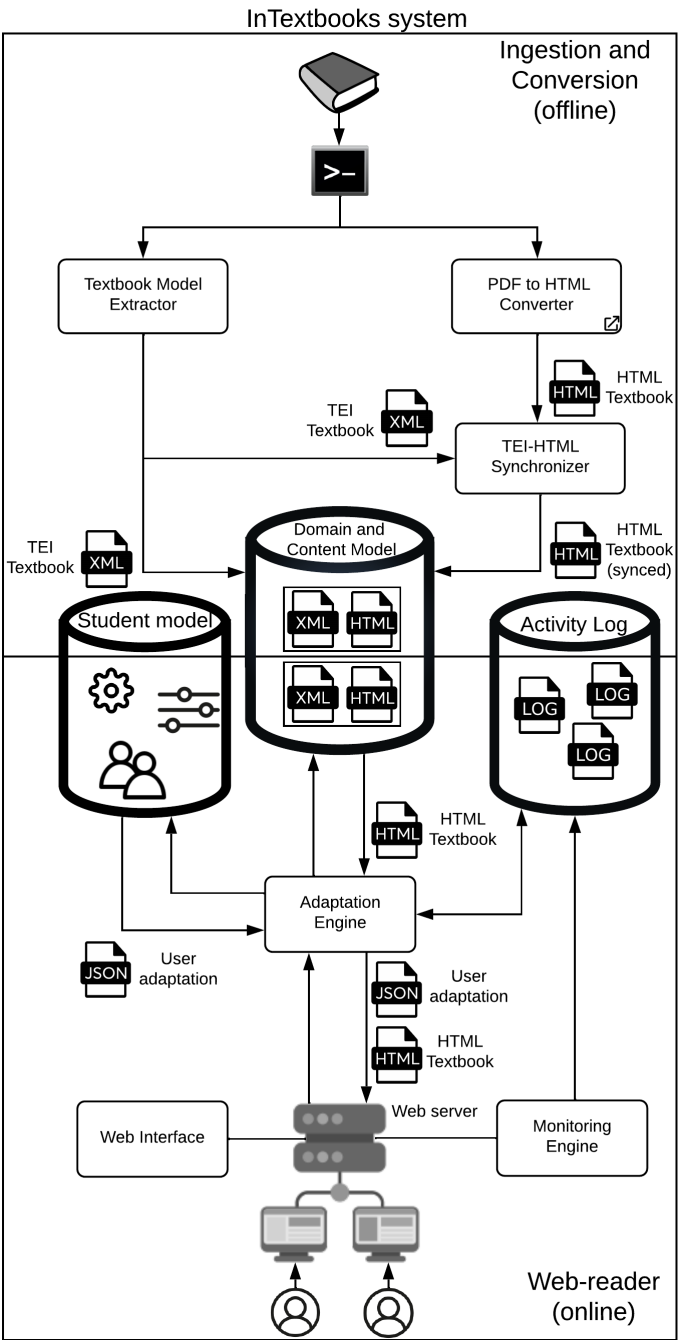


Figure 7.6: The InTextbooks architecture.

libraries were considered for this component: pdf2htmlEX⁴, PDFMiner⁵, pdf2html⁶, and Xpdf⁷. A comparison of the libraries was performed using different properties. The most important one was the ability of the conversion library to preserve the look of the PDF in the resulting HTML. Therefore, the 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. Linux support, performance, and scalability were other considered factors. After analyzing the candidate libraries, pdf2htmlEX was chosen. 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 knowledge 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 has a DOM structure that identifies words, lines, and paragraphs with the same IDs from their corresponding elements in the TEI document. The next three sub-paragraphs explains the matching algorithm to synchronize both representations of a textbook.

– **Internal Representation.** The matching algorithm 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, extra information is kept, such as their X- and Y-position on a page and font size, which are necessary for the consequent matching. For the words, the algorithm keeps 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, there is a list of the *TextNodes* that all together form a word in the HTML according to the TEI model.

⁴<https://github.com/coolwanglu/pdf2htmlEX>

⁵<https://pypi.org/project/pdfminer/>

⁶<https://github.com/mgedmin/pdf2html>

⁷<https://www.xpdfreader.com/pdf-to-html-man.html>

– **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 (🔗 Chapter 3, Section 3.4.1). This library compares the words and it is used to determine 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, the algorithm 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 pre-processing step, some special Unicode characters are replaced by their correct counterparts. The replacement list is stored in a separate file; which is extended as new cases are observed.

– **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.

Web-reader

Web Interface. The web interface of the InTextbooks system will allow students to engage and interact with the textbooks in several ways. Currently, this and the other online components (e.g., the student model) of the system are in a design and development stage. Figure 7.7 presents the working prototype of the web interface. The interface is divided into three parts. The main one is shown in the middle; this is where the textbook is displayed. There are smaller panels on either side of the

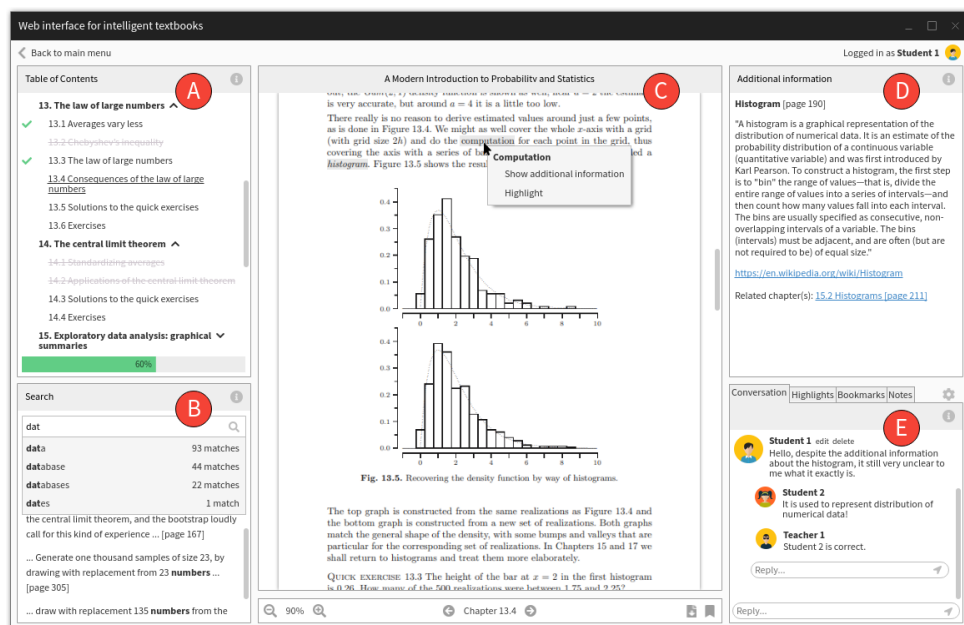


Figure 7.7: InTextbooks web interface prototype.

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 7.7). 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 entries 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 in Figure 7.7). 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 knowledge model of the textbook. Panel D in

Figure 7.7 shows the additional information for the *histogram* term. Other features are grouped under the same panel using tabs (panel E in Figure 7.7). 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, four tools have been defined, 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 a 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 in Figure 7.7). 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 the 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.

Monitoring Engine. The web interface communicates directly to the monitoring engine to log every action of a student. All the logs are stored 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.

7.3.4 Validation

A validation experiment was conducted 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, and adaptive interactions with textbooks.

Procedure

A test set of 70 university-level textbooks was used. The set contained textbooks in several domains: statistics, computer science, web programming, literature, and history. All textbooks were written in English. To estimate the accuracy of the matching algorithm, the percentage of words that were matched between the TEI and HTML representations of each book was used as the evaluation metric. Such 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.

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, another version of the matching algorithm was implemented. This version 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. The validation was carried out using three variations of the matching algorithm: the original matching algorithm (no threshold), the algorithm with a fixed threshold, and a variant with a dynamic threshold based on the most frequent distance between two lines on a page.

Results

Table 7.1 shows the results for the validation using the test set and the three algorithms. 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 [69], 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, the algorithm should be able to get an accuracy approaching the 100% mark.

Table 7.1: Validation of the matching algorithms.

Statistics	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

7.3.5 Discussion

InTextbooks is a novel system that transforms PDF textbooks into interactive and intelligent educational resources. The system’s goal is not only to offer a Web interface that will allow learners to interact with the textbooks in multiple ways but also to offer adaptation for each possible element. The technology of InTextbooks is possible since every content element (words, lines, text fragments, etc.) of the textbook

is individually identified in the knowledge models. This level of structural granularity makes it possible to create a fine-grained DOM structure in the HTML version of the textbook. The matching algorithm links around 88% of all the words in the textbooks to individual elements in the HTML resources. This precise identification of elements in both representations allows for adapting the content.

The designed web interface allows interaction with the HTML textbooks in multiple ways. Since every object of the textbook is identifiable in the Web interface, they can be linked to actions (e.g., searching and highlighting) or additional content (e.g., definitions and exercises). Additionally, each object can be shown or hidden depending on the user's needs.

The main challenge for the future of InTextbooks is to add the missing components of the system (e.g., student model) to infer the current state of student's knowledge and provide meaningful adaptation.

7.4 InTextbooks/P⁴: Integrating Textbooks with Smart Interactive Content for Learning Programming

7.4.1 Motivation

Electronic textbooks and various kinds of "smart" interactive systems such as ITS or virtual labs have been traditionally considered as two opposite ways to leverage the power of computers for human learning. The research on electronic textbooks attempted to enhance learning-by-reading supported by traditional textbooks by augmenting them with internal hyperlinks [223], semantic references [7], links to external material [117], annotations [167, 288], and even question answering [56]. In contrast, interactive learning tools focused on supporting learning-by-doing by offering students a chance to solve problems with an assistance of an intelligent tutor [19, 286], examine interactive worked examples [170, 298], or explore simulations [207].

Gradually, the recognition of complementary nature of learning-by-reading and learning-by-doing encouraged an increasing stream of research on integrating textbooks with interactive content [45]. This work has been most noticeable in computer science domain, such as learning programming languages, where researchers and practitioners developed and explored a broad range of "smart" interactive learning content [41] such as coding problems [85], interactive examples [298], Parson's problems [216], and program visualizations [207]. Starting from the early attempt to augment multimedia and Web-based programming textbook with live problems [36], intelligent tutors [44], and interactive animations [37], the research on programming textbooks with "smart" interactive content led to the development of modern online interactive textbooks that are used by thousands of users [88, 245].

Yet, textbooks with "smart" interactive content are still a minority among other types of learning tools due to considerable problems of integrating traditional text with "smart content". While technical problems associated with integration are being gradually addressed by modern interoperability standards, the conceptual prob-

lems related to linking text with interactive content (which interactive activity is the best match for a section or text?) are not yet resolved. The current generation of interactive textbooks is still developed by manual allocation of interactive content developed by the textbook authors to textbook sections. This approach has scaling problems and complicates the reuse of smart learning content created by other authors. While approaches for automatic linking of textbook sections with various types of text-based resources such as other textbooks [117] or Wikipedia [7] have already been developed, automatic linking of text and complex activities has not been attempted. This section presents a first attempt to cross the border between text and smart interactive content for learning computer programming. As a domain to explore linking text with interactive content, programming domains offer one substantial advantage. The well-structured nature of programming code associated with interactive content makes it possible to extract knowledge components from the code in a scalable way [131]. Specifically, the integration of two educational systems, InTextbooks and P⁴, has been performed on a novel ontology-based linking approach. The approach solves two types of automatic linking problems: augmenting textbook sections with smart interactive content and extending topic-focusing collections of smart content with relevant reading resources.

7.4.2 Background

Effective integration of various types of learning content and systems serving it has been both an important practical problem and a long-standing research challenge for the developers of educational software. On a more practical side, several issues have been addressed with different degrees of success. For example, there is a range of standard protocols for reliable identification of users across multiple systems [125, 132, 237]. Also, strong community support exists behind standards for learning record stores aggregating educational data from external sources [3, 137]. At the same time, several interoperability standards have struggled to reach wider adoption despite initial promises [2, 134, 136]. From the research perspective, the Artificial Intelligence in Education (AIED) community has explored the problem of integration of intelligent and adaptive educational systems on multiple levels, including distributed personalisation architectures [39, 267] and centralized student modeling servers [46, 148], mapping domain models [253], and educational ontologies [77]. Ultimately, the motivation for such integration is a composition of a richer, more effective educational environments that can provide guided access to an assortment of educational content of different types and enable deeper learning. Reading material is an integral component of such educational setups as the main source of conceptual knowledge and potential destination for reflective and remedial learning activity.

From the architecture perspective, there are two primary models for integrating textbooks with smart interactive content: linking external interactive content into the relevant parts of a textbook; and linking relevant fragments of a textbook into an existing interactive education system. The former method has been implemented in several successful systems. For example, the classic adaptive system for learning LISP, ELM-ART [44], is organized as an electronic textbook augmented with training exercises. Students reading the textbook can practice their knowledge, thus provid-

ing ELM-ART with evidence for student modeling and adaptation. Another example of a similar organization is Runestone textbooks [88] augmented with several types of interactive content including Parson's problems. Examples of the second method are less numerous. Sosnovsky et al. [254] describe OOPS, an adaptive service that recommends relevant sections from a textbook to students solving self-assessment quizzes on Java. It is worth noting that another important distinction of the OOPS service is that it linked textbook sections to relevant quizzes and questions in an automated way.

The automated linking of textbooks is another important stream of research. However, it was not possible to find other examples of automatic linking of textbooks to smart interactive content besides OOPS. Most authors have looked into different ways to either cross-link multiple textbooks within the same domain [117], or integrate textbooks with external repositories of reading material [7, 193]. The approach presented in this section seeks to fill this gap by implementing a linking model between textbooks and smart interactive programming content.

7.4.3 The Application

Systems

Two systems are used for the linking of textbooks and smart interactive programming content: InTextbooks and the Python Programming Personalized Practice System (P⁴). The first one was introduced in Section 7.3.

Python Programming Personalized Practice System. P⁴ is an online personalized system offering students in introductory Python programming courses to practice their skills using several types of interactive learning materials. The system is designed as a non-mandatory practice and self-assessment tool that each student could use for individual needs. P⁴ was developed by using the Mastery Grids system [116] as its core. Each topic within the Python course is represented by a square cell in the top row (top left in Figure 7.9). Students can monitor their progress by checking the color of the grid cells, i.e., the greener the cell the more correct activities they have within that topic. For accessing the learning materials on a specific topic, students have to click the corresponding topic cell, which opens the learning activities selection section (left center part in Figure 7.9). Several types of learning activities are presented here, all of them in a different row, ranging from "Animated Examples" to "Parson's Problems". On top of it, personalized guidance is provided, based on a concept-level model of student's Python knowledge. The conceptual structure of the student model is driven by an ontology of Python programming concepts⁸ (from now on, the Python ontology), which was homologically created by using a Java ontology as a template [131]. The ontology is composed by leaf and inner nodes. A leaf node represents a Python concept. Inner nodes are used as a hierarchy of classes for the concepts. The model is built by observing student behavior in the system and represents the probability of students knowing each Python concept. To make this learner model "open" to the student, it is visualized as a bar chart on the bottom part of the activity selection interface (Figure 7.9). Each bar depicts one concept, and the height represents the estimated level of knowledge (i.e., the taller

⁸<http://acos.cs.hut.fi/static/python-parser/ontology.png>

it is, the more the estimation of knowledge). Based on these concept estimations, P⁴ recommends the three learning activities that are most appropriate for the student at that stage by following a specific learning goal (e.g., knowledge maximization or misconceptions' remediation). Recommended learning materials are highlighted with stars of different sizes within the interface (Figure 7.9).

Intextbooks/P⁴ Integration. The application presented in this section primarily benefits from the annotation of the textbook's content with domain terms in InTextbooks and the topic-concept-activity model in P⁴. Each content unit (page, subchapter, chapter) is annotated with its corresponding domain terms in the resulting knowledge models for a textbook. When those domain terms are linked to the Python programming concepts used in P⁴, two potential integrations are enabled: (1) learning activities from P⁴ can be displayed along with the corresponding content units in InTextbooks, and (2) content units from textbooks in InTextbooks can be additional learning activities associated with the most appropriate topics in P⁴. Here, the two-way integration for learning programming is presented in each of the two systems. As a working example, the content from the "*Python for Everybody*" textbook [244] has been linked with the topics-concepts-activities in P⁴. Specifically, the following examples show the link between a subchapter from the textbook and the "While Loops" topic (Figures 7.8 and 7.9). Figure 7.8 reflects the addition of learning activities associated with specific subchapters in InTextbooks. Since each subchapter is annotated with domain terms, learning activities from P⁴ that cover the same conceptual terms can be included in InTextbooks. When a user is navigating a subchapter linked with one or more learning activities, these are displayed as additional content (top-right panel in Figure 7.8). The user can interact directly with the learning activities without leaving the system. Figure 7.9 shows how "Textbook readings" were added as an additional type of learning activity (last row) in P⁴. When a "Textbook readings" cell is clicked, P⁴ directs students to the Reading Mirror system [25] (online reading tool integrated within P⁴), specifically focusing on the corresponding subchapter that has been associated with the topic given the concepts it covers. In the same way, as the other types of learning activities, "Textbook readings" are capable of being recommended to students as well (specially when the concepts covered were just introduced or need to be reinforced).

Integration Methodology

The goal of this integration is to map the knowledge models extracted from Python textbooks to the concepts and topics model available in the P⁴ system. This mapping allows for the interchanging of content from both models: (1) learning activities from P⁴ can be displayed directly in the InTextbooks system; and (2) subchapters from the Python textbooks that present and discuss the topics used in P⁴ can be displayed as an additional type of content. The methodology for mapping both models include three main steps and two sub-steps:

1. Knowledge model extraction and glossary unification
2. Glossary - ontology linking
3. Granular content linking
 - 3.1. Textbook subchapters to P⁴

The screenshot displays the InTextbooks/P⁴ interface. On the left, a 'Table of Contents' sidebar lists various topics, with '5 iteration' expanded to show sub-topics like '5.1 updating variables', '5.2 the while statement', and '5.3 infinite loops'. A 'Search' bar is also present. The main content area shows the textbook page for 'Chapter 5: Iteration', specifically '5.1 Updating variables'. The page includes a title, a brief introduction, and a code example for updating a variable. On the right, a 'Learning activity #1' panel provides an example of a program that receives an integer and stops receiving more integers when the user enters a negative integer. Below this, there are links for 'Learning activity #2', 'Learning activity #3', and 'Learning activity #4'. A 'Notes' panel at the bottom right states 'the while statement is used for iteration'.

Figure 7.8: Integration of external learning activities as an additional content within InTextbooks, given the automatic linkage between textbook domain terms and Python programming concepts.

The screenshot displays the P⁴ interface. On the left, a 'My Progress' section shows a progress bar for various topics, with '5.2 the while statement' highlighted. Below this, a 'Recommended Activities' section lists activities like '1. while loop with summation', '2. iteration with Addition', and '3. Comparing Adjacent Numbers in a Sequence of Numbers'. A 'Current topic: While Loops' section shows a progress bar for '5.2 the while statement' with a progress of 0%. On the right, a 'Recommended Activities' section lists activities like '4 functions', '5 iteration', and '6 strings'. A 'Current topic: While Loops' section shows a progress bar for '5.2 the while statement' with a progress of 0%. The main content area shows the textbook page for 'Chapter 5: Iteration', specifically '5.1 Updating variables'. The page includes a title, a brief introduction, and a code example for updating a variable.

Figure 7.9: Integration of textbook sections as an additional learning material within P⁴, given the automatic linkage between domain terms and Python programming concepts.

3.2. P⁴ learning activities to textbook subchapters

Figure 7.10 shows the elements from both systems (InTextbooks and P⁴) used in the methodology. Each step is described in the following paragraphs.

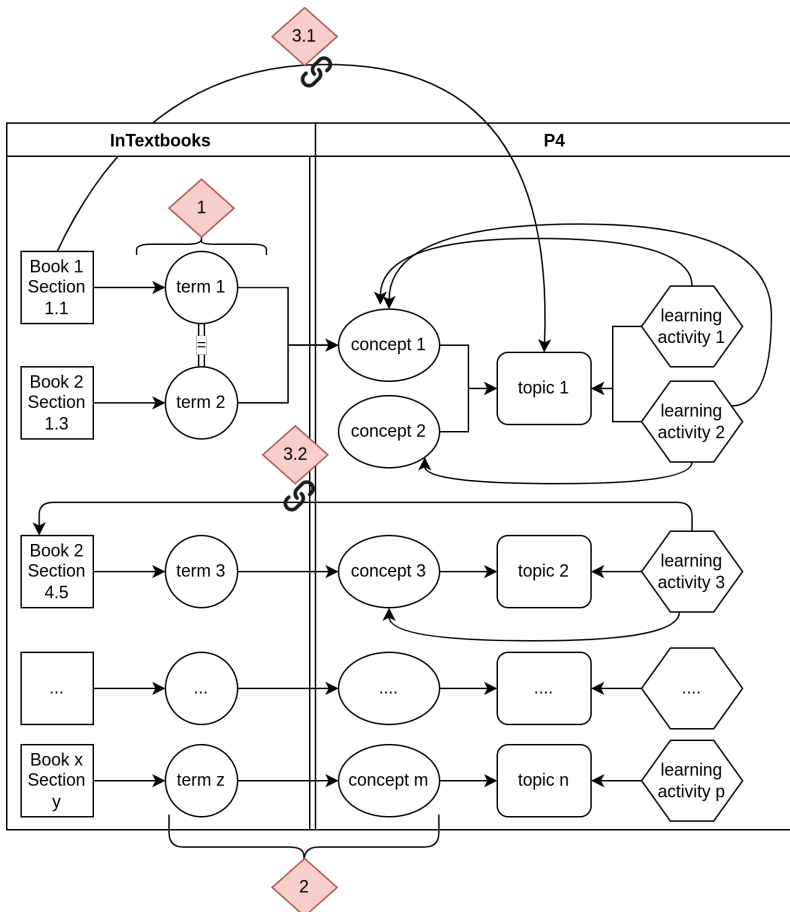


Figure 7.10: Elements from both InTextbooks and P⁴ used in the different steps of the methodology.

Knowledge model extraction and glossary unification. The first step in the methodology is to extract the list of all the index terms present in the Python textbooks to create a single unified glossary of terms to be mapped with the Python ontology. First, for each textbook, a knowledge model is extracted and enriched with semantic information from DBpedia using the approach presented in this thesis (🔗 Chapters 3 and 4). For the enrichment process, the DBpedia category *dbc:Computer_programming* is used. Then, a glossary of index terms is created for each textbook. Since two terms representing the same concept can be written differently, all glossaries are merged to identify repeated terms (🔗 Chapter 4, Section 4.3.5). For example, a term in the finally glossary is {PF: *if statement*; AL: *statement () if*; EC:- }. PF is the preferred label, which corresponds to the ID of the term. AL is a set of alternative labels, which includes all the identified versions of the same term. EC is any linked external concept (e.g., a DBpedia entity). Initially,

no term has an external concept associated with it; this will be done in the next step.

The result of this step is one unified glossary with the index terms from all the textbooks. The number 1 in Figure 7.10 illustrates that the unification of the different terms from the textbook forms the unified glossary.

Glossary - ontology linking. The next step in the methodology is to link the terms from the glossary to the concepts in the Python ontology. First, the concepts in the Python ontology are extracted. For each concept, both the single name (e.g., *while*) and the compound name with the parent classes (e.g., *while* () *iteration statement* () *statement* () *python language* () *python*) are retrieved. Then, a linking strategy is applied. First, glossary terms are linked to the Python concepts using exact textual matching between the different labels of the terms and the single name of the concepts. Then, stemming is applied to find more matches. Third, a small list of abbreviations is used. The list was created manually since the Python ontology uses some abbreviated names (e.g., *int* and *div*) that appear with a full name in the glossary (e.g., *integer* and *division*). Fourth, the parent classes of the Python concepts are used. Since the class hierarchy in the Python ontology has a deeper granularity than the names used in the textbooks, matches between a complete glossary term and a partial compound concept name are accepted. For example, the glossary term *if statement* is linked to the concept *if* () *selection statement* () *statement* () *python language* () *python* since the term matches the concept's single name (*if*) and one of its parent classes (*statement*). Finally, the glossary terms that are not linked to any Python concept are compared to the inner classes of the Python ontology (e.g., *Boolean Expression*) using the mentioned strategies (textual matching, stemming, and synonyms). The result of this step is that the linked glossary terms have a Python concept as an external concept (e.g., {PF: *if statement*; AL: *statement* () *if, statement* () *conditional*; EC: *if* }). The number 2 in Figure 7.10 illustrates the linking between the terms in the glossary and the concepts from the Python ontology used in P⁴.

Granular content linking. The final step is to link the textbooks from InTextbooks with the learning activities from P⁴.

– **Textbook subchapters to P⁴.** First, a list of the concepts introduced in each topic is extracted from P⁴. Then, for each textbook, the linked glossary terms to Python concepts are used to get the individual index terms in the textbook associated with the external concepts. After that, the subchapters associated with the index terms are retrieved from the knowledge models. Now, there is a map from subchapters to index terms, from index terms to concepts, and from concepts to topics. Finally, using the concepts as a bridge, each subchapter is linked to the topics in P⁴ where the Python concepts are introduced. This sub-step is represented with the number 3.1 in Figure 7.10.

– **P⁴ learning activities to textbook subchapters.** In P⁴, each learning activity has a set of associated concepts, plus the topic where it is used. First, this list of learning activities is extracted and then, the topics that are prerequisites using the associated concepts and the topics where they are introduced are computed. After

that, for each subchapter linked to a topic in P^4 , the learning activities that belong to the same topic and have one of the concepts associated with the subchapter are selected as candidates. Then, there is a check to see if the topic prerequisites for each learning activity have been introduced in other previous subchapters. If all the prerequisites are met, a subchapter-learning activity pair is created. This sub-step is represented with the number 3.2 in Figure 7.10.

Results

The proposed methodology was applied using 5 different Python textbooks ([80, 120, 149, 244, 276]) and the mentioned Python ontology. In this subsection, the obtained data for each step of the methodology are described and analyzed.

Knowledge model extraction and glossary unification. After creating the knowledge models for the textbooks, there was a variate number of index terms in each one (817, 848, 834, 451, 1105), producing a total of 4055 different elements. After merging the terms, the unified glossary contained 3250 elements (a reduction of almost 20%). Additionally, when processing the Python ontology, 108 elements were obtained: 73 leaf concepts and 35 inner classes.

Glossary - ontology linking. There were 53 instances after linking the terms in the glossary to the elements in the Python ontology. Some terms in the glossary were linked to the same Python concept. Thirty-six concepts and ten classes from the Python ontology were linked to the 53 terms in the glossary.

Granular content linking. At the final step of the methodology, there were multiple contents linked in both systems. First, 266 different index terms from the five textbooks were linked to 217 concepts and 49 classes from the Python ontology. Of those 266 terms, only 186 were linked to concepts used in P^4 , since not all the Python concepts from the ontology are currently a part of the system. Using the linked index terms and the Python concepts, 245 different subchapters from all the textbooks were mapped to the topics in P^4 . The number of linked subchapters for each topic is as follows: 36 to "Variables and Operations", 35 to "Boolean Expressions", 12 to "If-Else", 3 to "While Loops", 19 to "For Loops", 40 to "Functions", 49 to "Lists", 11 to "Dictionary", 17 to "Strings", 3 to "File Handling", 5 to "Exceptions", and 15 to "Classes Objects". Regarding linking learning activities to subchapters, 2240 possible mappings were analyzed, from which 1790 fulfilled the prerequisite restrictions. If each learning activity is only assigned once in the whole textbook, instead of to multiple subchapters, the average of unique learning activities mapped per textbook is 67, from a total of 157 different activities available in P^4 .

General Analysis. The obtained data show that despite the small number of glossary terms that are matched to Python concepts (53), the number of both linked subchapters to topics (245) and the learning activities to subchapters (1790) is promising. P^4 can benefit from incorporating textual material explaining the concepts used in each topic. Additionally, multiple textbooks enable more personalization: a student can select one or more textbooks to get the textual material recommendations according to their preferences. InTextbooks can present additional interactive content to the learners as they progress and navigate through a textbook. Currently, the same learning activity is linked to all the fitting subchapters within a textbook, which will cause problems if the textbook is read sequentially. This approach has

been used because InTextbooks could generate personalized navigation paths for each student. Hence, it is helpful to have an extensive mapping of learning activities. The system will need to be aware of this situation and not display learning activities that have been seen already in previous subchapters. Finally, the order of topics and the association of concepts to learning activities in P⁴ allows the comparison of prerequisite-outcome relations with the textbooks. The order of topics in four of the five textbooks was similar to the one in P⁴. However, one textbook [276] was an exception, producing no links to learning activities since the required prerequisites were not introduced before, or not at all, in the textbook. Since the purpose of the textbook was not to teach Python but to use it for data science, the author assumes familiarity with the language, resulting in fewer programming concepts being introduced.

7.4.4 Validation

In order to assess the automatic concept linking approach described above, two domain experts with experience in teaching Python independently rated the match quality (appropriateness) of the textbook sections conceptually linked to each of the topics presented in the P⁴ system. The following rating schema was used: 3 as a good match, 2 as a partial match, and 1 as a bad match. 55 sections of the "*Python for Everybody*" textbook were automatically associated to one of the 17 topics of the Python course. Note that the topical structure of that Python course was defined by Python instructors from several universities, who used P⁴ systems in to support Python practice in their courses.

After the scoring, the results were examined to determine possible causes for the discovered bad matches. The examination revealed that a number of low matching scores were produced by the sections titled as "Glossary" that were included in each chapter rather than assembled at the end of the book in a more traditional way. The nature of these sections make them poor independent learning resources since they served as a reminder of already learned content. It was decided to exclude these sections from matching and evaluation.

After glossaries were removed, inter-reliability between raters was calculated using weighted Cohen's Kappa. The resulting Kappa 0.63 is considered as moderate inter-reliability. Given that raters did not have made ratings fully independently and that they only had access to a very short and concise description of the rating schema, this result is deemed as positive. The main disagreement was registered in the topics of "Boolean Expressions", and "Lists/Strings". The point of disagreement here was that in the textbook some chapters introduce some concepts blended together (e.g., *boolean operators* and *if statement*) while in the P⁴ course they are clearly separated and taught one after another (i.e., first *boolean expressions*, and then *if-else*). In terms of general mutual agreement, both raters coincided in thinking that there was a good match in 45% of the textbook associations, a partial match in 12% of the cases and a bad match in 14% of the total linkages. Considering the evaluations that lead to disagreement (29%), 17% involved positive evaluations (either one of the two ratings as 3 or 2), while only a 12% lead a bad rating for the match. Finally, in total, 74% of raters' pairs of evaluations were at least either partial or suitable matches, so it is possible to conclude that the proposed automatic

linking approach leads to acceptable results as a first step to resolving the problem of automatic linking.

7.4.5 Discussion

The InTextbooks/P⁴ integration is a two-way linking between textbook sections and smart learning content items generated by the automatic extraction of concepts from programming textbooks using a combination of textbook knowledge models and different ontologies as underpinning tools for this task (e.g., DBpedia, Python ontology). This integration is a first step to resolving the problem of automatic linking. Thanks to the knowledge models, the index terms can be connected to the Python ontology, which creates a two-way integration. The content from the textbooks is a valuable resource to be displayed in P⁴, and InTextbooks benefits from the "smart" interactive content that can be displayed when students have read the appropriate sections. This integration shows that the domain representation obtained from merging knowledge models from multiple textbooks is functional when linking content. Additionally, this application shows that the automated linking of textbooks to external content is possible when the necessary knowledge is available in a machine-readable format.

Given that the textbook concept extraction works by analyzing the textual content, a natural next step would be exploring a more "fine-grained" association of concepts, e.g., at a paragraph level rather than on a section level. This approach will enable practice learning material to be recommended to the users "in context", right after the student reads the corresponding lines where a concept is introduced. In a similar way, given that the textual information presented in the textbook generally focuses on presenting the concepts in an introductory way, a more "fine-grained" association in terms of textual units will enable the systems to recommend remedial reading when students fail in specific learning activities which could reflect the learner's misconception(s) of certain concept(s).

7.5 Conclusion

This chapter presented three different AIES built on top of the extracted textbook knowledge models described in this thesis. Each of them benefits in its own way from the knowledge encoded in the textbook models. The potential uses of the extracted knowledge models are many since the connections between the content and the models can be leveraged in multiple ways.

As an additional example, Dresscher et al. [81] experimented with the automated generation of assessment questions using the extracted textbook knowledge models. In this application, the enriched knowledge models provided the textual content (textbook sentences and information from DBpedia) necessary to generate the questions.

For all show-cased systems, one concern needs to be addressed: the availability of textbooks and copyright issues. 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 some publishers do not have a reason to object. For example, several publishers consented to use their textbooks in the research presented in this thesis. In the worst-case scenario, many good-quality textbooks are freely available online nowadays in open repositories such as Connections⁹, Open Textbook Library, OER-Commons¹⁰, etc.

⁹<https://cnx.org/>

¹⁰<https://www.oercommons.org/>

CHAPTER 8

Conclusion

This thesis has explored and presented an approach for the automatic extraction of knowledge models from digital textbooks. Chapter 2 described different approaches for knowledge extraction that can be applied to textbooks. Chapter 3 presented the proposed workflow to extract knowledge from textbooks taking into account their formatting rules and internal structure. Chapter 4 described connecting the knowledge models to the Linked Open Data Cloud to get additional semantic information. Chapter 5 detailed the categorization of the index terms according to their relevance to the target domain. Chapter 6 presented a validation experiment where the concepts extracted from textbooks were recognized as valid units for knowledge modeling. Finally, Chapter 7 presented three systems designed with the extracted knowledge models at their core. A brief description of the tools implemented using the proposed approach is presented in Appendix B.

In this final chapter, the research questions are revisited and discussed. Additionally, limitations and future work are discussed.

8.1 Revisiting the Research Questions

In the Introduction (📌 Chapter 1, Section 1.4), the main research question was formulated:

RQMain Can high-quality and domain-specific knowledge models be automatically extracted from textbooks?

Eight subquestions were posed to help answering the main question. This section lists and answers these questions. Together, they form an answer to the main research question.

RQ1 What are the characteristics of existing approaches to extract information elements from textbooks?

Different approaches have extracted information elements from textbooks across three categories: content (e.g., layout and content objects), structure (ToC), and domain (e.g., terms and topics). Different methods have been used (e.g., rule-based, NLP-based, learning-based, and knowledge-based) with the support of different external sources (e.g., Wikipedia and Web searches). The extracted information has been represented using different formats (e.g., XML and ontologies languages). However, there is a lack of methods that extract information across all three categories and encode all the information using a human and machine-readable format. The proposed approach in this thesis fills in this gap to extract full-scale knowledge models that can be used for multiple applications (e.g., textbook linkage).

RQ2 Can the structure, content, and domain terms be automatically extracted from textbooks, and if so, what is the *accuracy* and value of the extracted information?

The proposed approach describes multiple steps to automatically recognize the structure, content, and domain terms from textbooks. The structure of the textbooks is extracted using the Table of Contents. Content for each section (chapter

or subchapter) is extracted by recognizing each textual element using a bottom-up approach: first words, then lines, then fragments, then pages, and, finally, sections. Domain terms are extracted directly from the back-of-the-book index. An evaluation showed that the structural elements (heading, hierarchy level, and page number) are recognized with almost absolute precision and recall in all the tested domains (*statistics*, *computer science*, *history*, and *literature*). The content is recognized with high accuracy, achieving better results than other state-of-the-art tools. Domain terms (heading and hierarchy level) are recognized with very high precision and recall (around 98%). The extracted textbook information is a valuable resource for knowledge-driven tasks. For example, an experiment showed that the extracted information could be used to cross-link relevant sections between different textbooks more effectively than traditional techniques (TF-IDF and LDA).

RQ3 Can the domain terms extracted from textbooks be linked to their corresponding entities in a global reference model, and if so, what are the *semantics* obtained from the linkage?

In the second phase of the proposed approach; after the structure, content, and domain terms have been extracted; multiple steps link the domain terms to entities in DBpedia. The linking mechanism finds possible candidates for each term, and then it applies a disambiguation technique to find the entity that matches the term's actual meaning ("sense"). An evaluation in the *statistics* domain revealed that the linking strategy can achieve high and balanced precision and recall values (e.g., 97% and 92%, respectively). The domain terms linked to their corresponding entities are enriched with additional information from DBpedia: abstracts, Wikipedia links, categories, and direct connections (relations) to other terms.

RQ4 Can the domain terms extracted from multiple textbooks be integrated into a single model, and if so, what is the *coverage* of such an integrated model?

The third phase of the approach describes integrating the domain terms from multiple textbooks into a single model by merging semantically equal terms. When more textbooks from the same domain are integrated, the coverage of the domain grows significantly. An evaluation showed that integrating glossaries from ten *statistics* textbooks achieved a more complete representation of the domain knowledge (one-third of the target domain) than using only single textbooks (less than one-tenth of the target domain). Additionally, another evaluation showed that with every additional *statistics* textbook used, the number of *in-domain*+ concepts increases. Finally, a validation study demonstrated that three *Python Programming* textbooks cover all the domain's essential concepts to perform domain modeling.

RQ5 Can the domain relevance of concepts extracted from textbooks be established, and if so, what is the *specificity* of such concepts?

The fourth phase of the approach details five steps to identify the relevance of concepts to the target domain. The information from DBpedia (categories, links, and abstracts) and textbooks (index terms and domain information) is used to identify how individual concepts are related to the target domain. The domain specificity of the terms is described using four categories: most important concepts in the target

domain, other concepts in the target domain, concepts in neighboring domains, and concepts unrelated to the target domain. An evaluation showed that distinguishing between the concepts relevant and non-relevant to the target domain is achieved with high accuracy (92% in the *statistics* domain). In the domain of *ancient philosophy*, the accuracy for multi-boundary detection (target, related, and unrelated domain) was 21 points higher than the baseline (72%).

RQ6 Are the concepts extracted from textbooks *cognitively valid* components for knowledge modeling, and if so, what is the *granularity* of such concepts?

A validation experiment analyzed the cognitive validity of textbook concepts for knowledge modeling using learning curve analysis. The results established that, in general, the analyzed concepts in the domain of *Python programming* are cognitively valid knowledge units. For 44 out of the 46 studied concepts, the learning happening during student practice followed the power law (the mean fit is 72%). In terms of granularity, textbooks contain both fine- and coarse-grained concepts. Both types can be used for knowledge modeling, but the analysis showed that fine-grained concepts model knowledge better.

RQ7 Is knowledge extraction effective across multiple domains?

The proposed approach was evaluated using textbooks from different domains. Extraction of the textbook content (structure, content, and domain terms) was accurate in the domains of *statistics*, *computer science*, *history*, and *literature*. *Statistics* and *information retrieval* terms were linked correctly to their respective entities in DBpedia. The specificity of terms in *statistics* and *ancient philosophy* was successfully identified. Finally, concepts extracted from *Python programming* textbooks showed to be valid knowledge components. Across all tested domains, the proposed approach was effective.

RQ8 How can the knowledge models extracted from textbooks support AIES?

Different evaluations showed that the extracted knowledge models have multiple applications—for example, semantic bridge, semantic search, and knowledge modeling. Three design exercises provided insight into the use of the models for developing AIES. The knowledge models act as a language bridge between textbooks written in different languages in Interlingua. Intextbooks supports the adaptation and interactivity of all possible elements in HTML textbooks thanks to the fine-grained identification of content elements in the extracted knowledge models. Finally, the integration of Intextbooks with P^4 shows how the knowledge models can act as a semantic bridge between textbook content and smart learning activities.

8.2 Limitations

The proposed approach extracts information from textbooks using their formatting rules and internal structure. One concern is how well the approach will cope with different textbook formatting. Naturally, formats can be quite different between different textbooks. Nevertheless, the authors and the publishers make every effort to ensure they are consistent within textbooks (otherwise, they would cause un-

necessary confusion). The described approach is built on the assumption that the formatting patterns of textbooks do follow a set of logical rules. The implemented rule-based workflow tries to capture these patterns. It has been observed that the variability is quite manageable, and as more and more textbooks are processed, the set of rules has to be updated progressively less. Having said that, the proposed approach has several limitations that still need to be addressed.

First, a knowledge model can only be extracted from textbooks having a back-of-the-book index. This situation restricts the pool of textbooks that can be used with the approach. Additionally, since the proposed approach relies entirely on the index as the source of domain terms, textbooks with a poor index will produce low-quality knowledge models.

Second, in the linking/enrichment phase of the proposed approach, domain terms are linked to entities in DBpedia. However, not all terms can be linked since some do not have a corresponding entity, or the approach can not find it. Many multi-level index terms refer to particular constructions (e.g., *histogram* {} *computed for* {} *exponential data*) that cannot be matched. Since the integration and categorization stages rely on terms matched to DBpedia, a low term linking affects the amount of knowledge contained in the models (semantic information, integration of terms, and domain specificity).

Third, the proposed approach processes the structure of the ToC and the index sections very well, but no meaning is extracted from them. The approach is not identifying more hierarchical relations and semantics (e.g., topics).

Fourth, even though the text extraction accuracy is high, there is content that the approach needs to process better. Formulae, tables, and captions, among other elements, are not recognized properly.

Finally, this thesis evaluated the proposed approach with multiple but formal domains. The approach's applicability in other, less formal domains (e.g., *medicine*), domains with low consensus or conflicting viewpoints (e.g., *history*), or dynamic and new domains (e.g., *xenobiology*¹) still needs to be explored. These domains could seriously challenge the proposed approach. For example, the knowledge models could suffer from high subjectivity in such domains.

8.3 Future Work

Future research is planned in three main directions: (1) further extension and evaluation of the proposed approach; (2) evaluation of extracted knowledge models from the prospect of pedagogical usefulness; and (3) extension and use of the approach towards creating intelligent textbooks.

The proposed approach should be extended with new functionalities. It is necessary to explore keyword extraction techniques to extract the domain terminology when an index is not present or to discover more relevant terms when the textbooks have a poor index. To increase term linking to external entities, more knowledge bases in addition to DBpedia could be used. The use of domain-specific thesauri, if available, can also increase the linking (e.g., The Medical Subject Headings (MeSH)

¹The science of estranged life forms.

thesaurus² for health-related information). Using more computational NLP (e.g., deep language models) will allow for extracting more information, such as learning objects (e.g., definitions and examples) and topics. Also, adding more relations between elements in the models is necessary—for example, prerequisite-outcome relations among concepts. Better content processing and utilization (e.g., formulae, tables/plots) are important because those elements are a source of semantics. Another task is further evaluating the approach using textbooks in a diversity of domains. Finally, testing the approach in less formal, dynamic, and new domains is necessary. The evaluation of the resulting models in terms of quality should also be extended. For example, this thesis did not explore the levels of subjectivity in the models.

This thesis has focused on the development of technology for the extraction of knowledge models from textbooks. As discussed, the extracted knowledge model can support the development of educational systems in multiple ways. The next logical step is to perform classroom studies to measure the pedagogical value of systems constructed with the extracted models.

Finally, the proposed approach could be further exploited to transform static textbooks into a new generation of intelligent textbooks. One direction is to create fully interactive HTML elements from the textbooks' static tables, charts, and formulae. Building a range of intelligent services on top of the extracted knowledge models is crucial. Fine-grained element adaptation, automatic text summarization, and semantic search are examples of such services.

²<https://www.nlm.nih.gov/mesh/meshhome.html>

APPENDIX A

Learning Curves

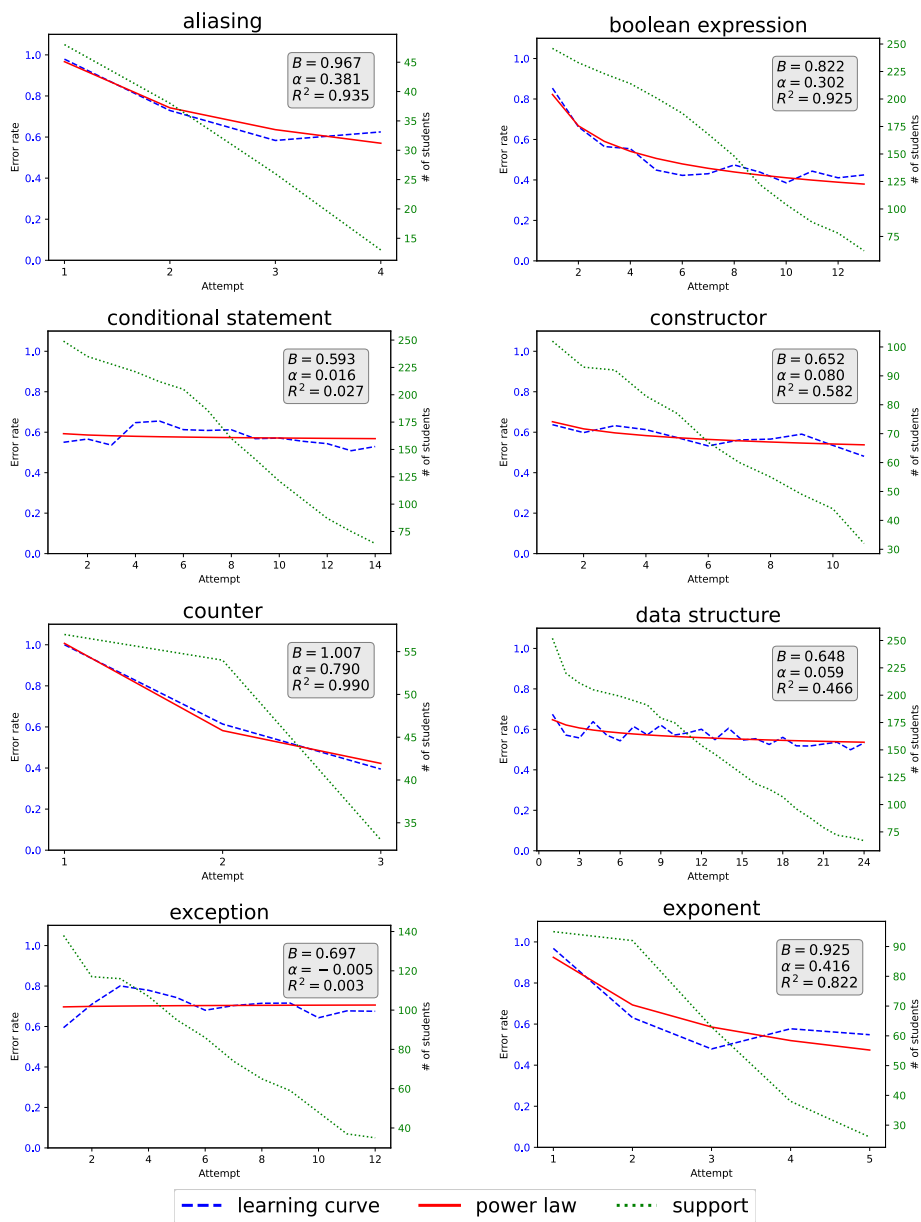


Figure A.1: Learning curves (1/6).

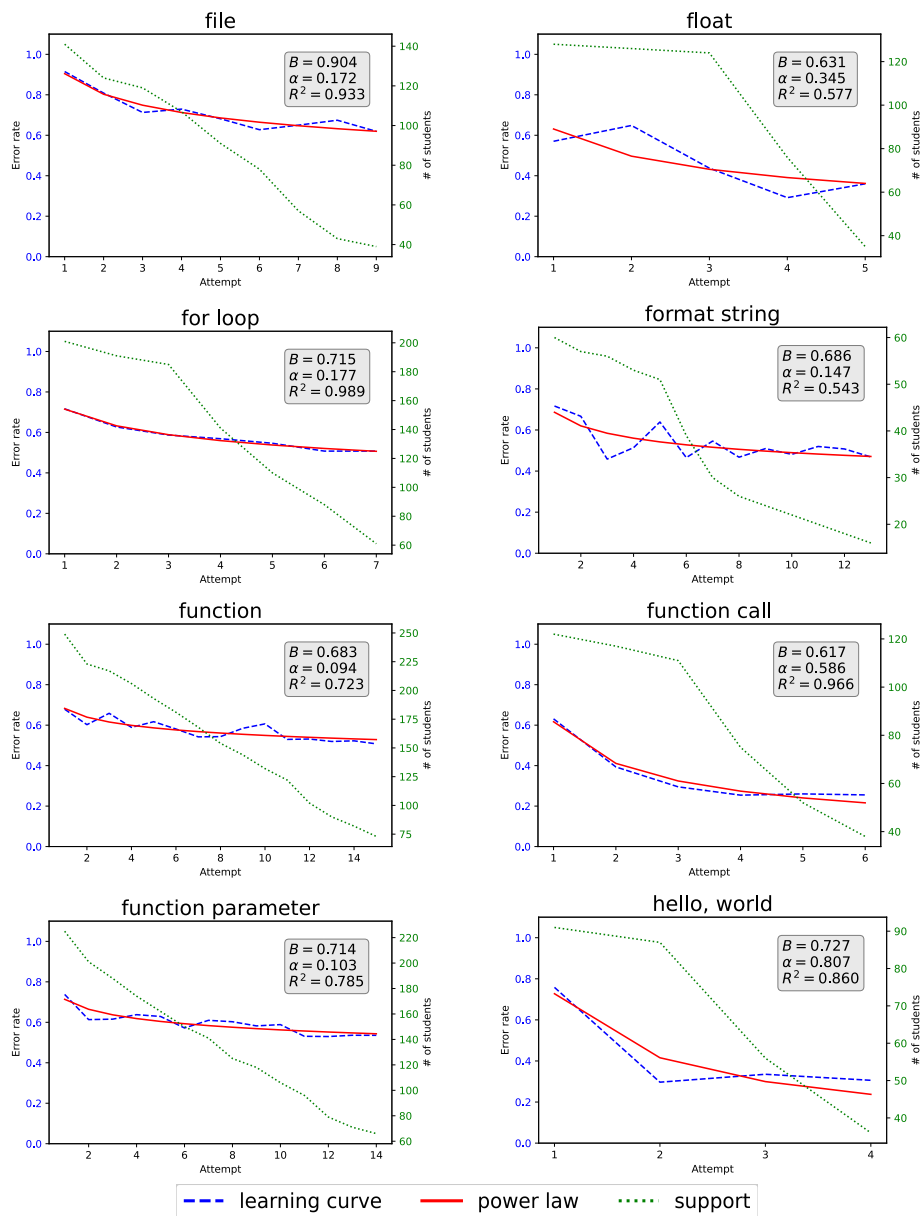


Figure A.2: Learning curves (2/6).

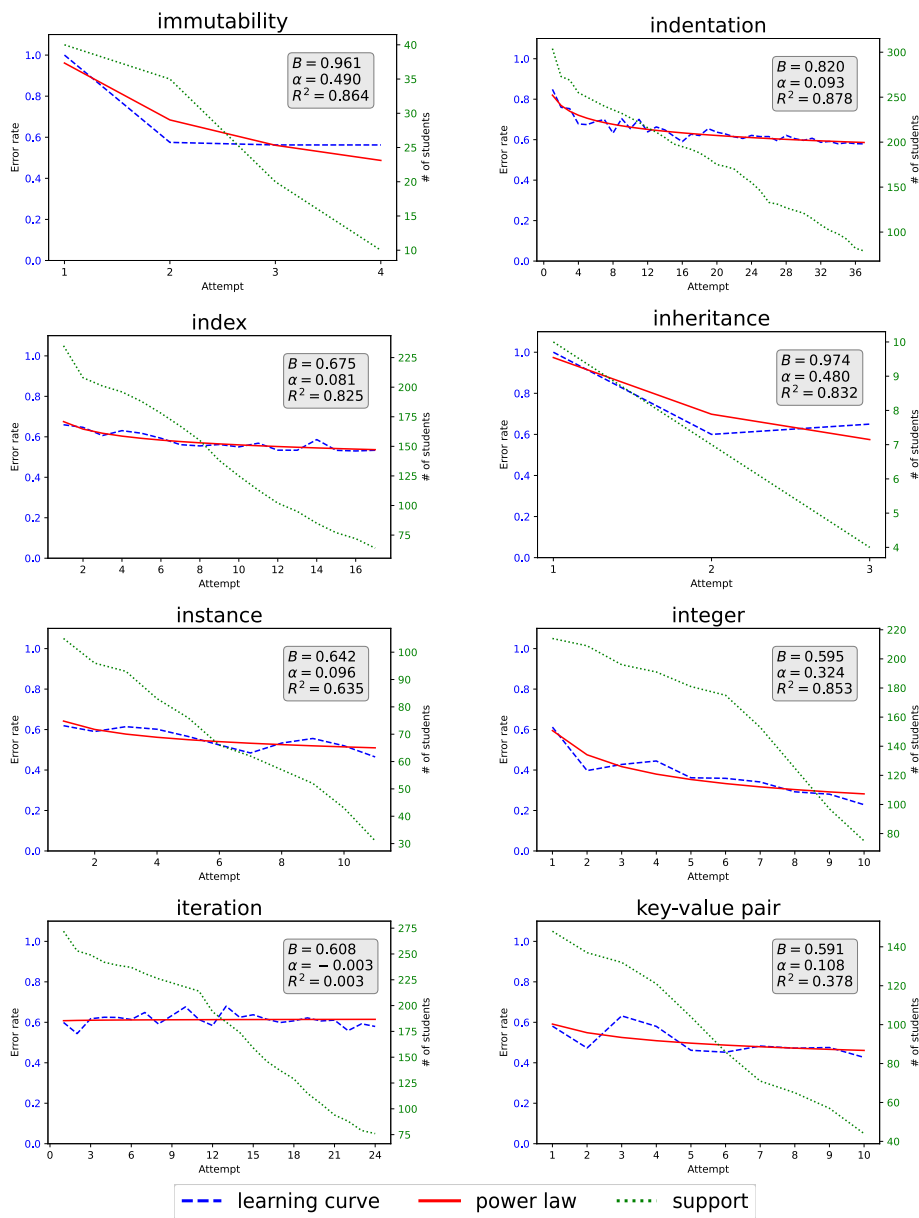


Figure A.3: Learning curves (3/6).

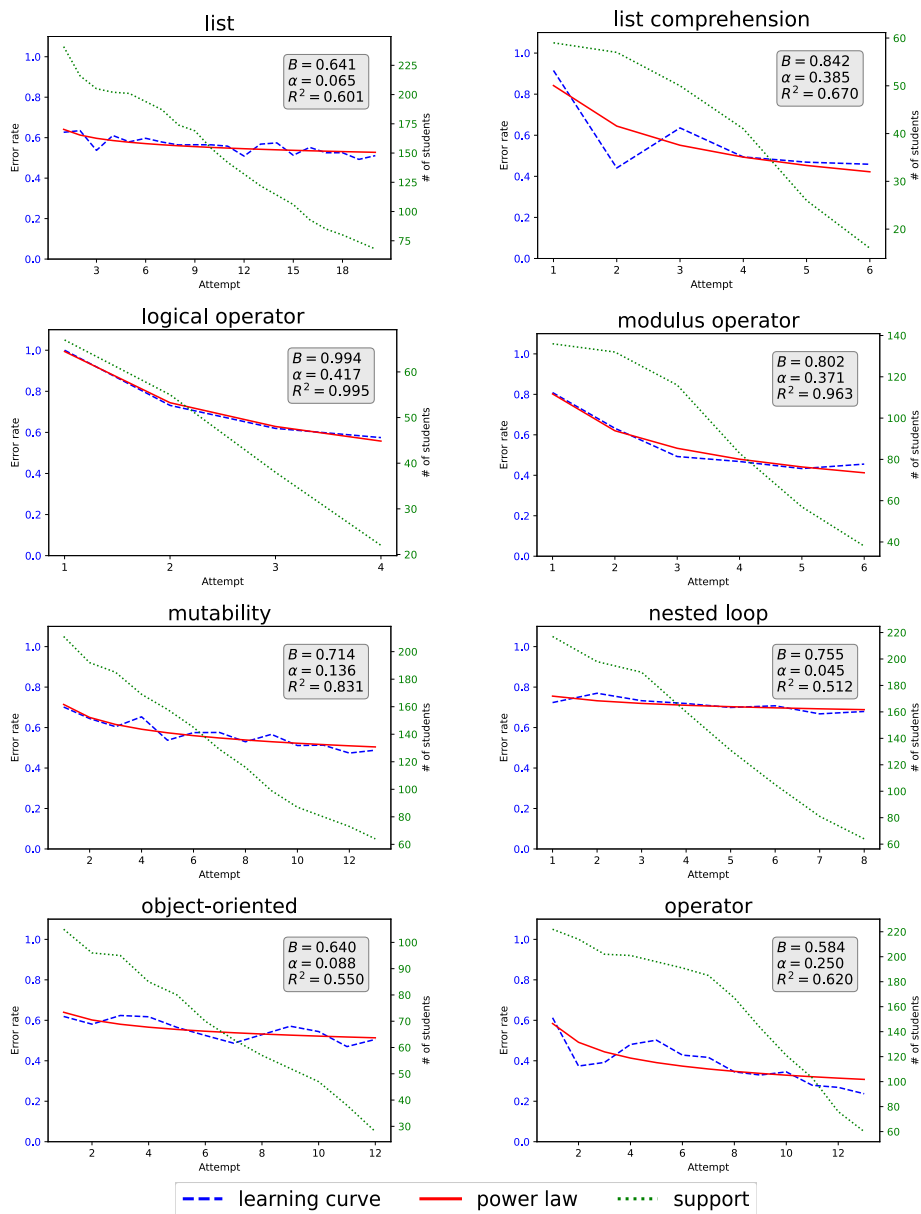


Figure A.4: Learning curves (4/6).

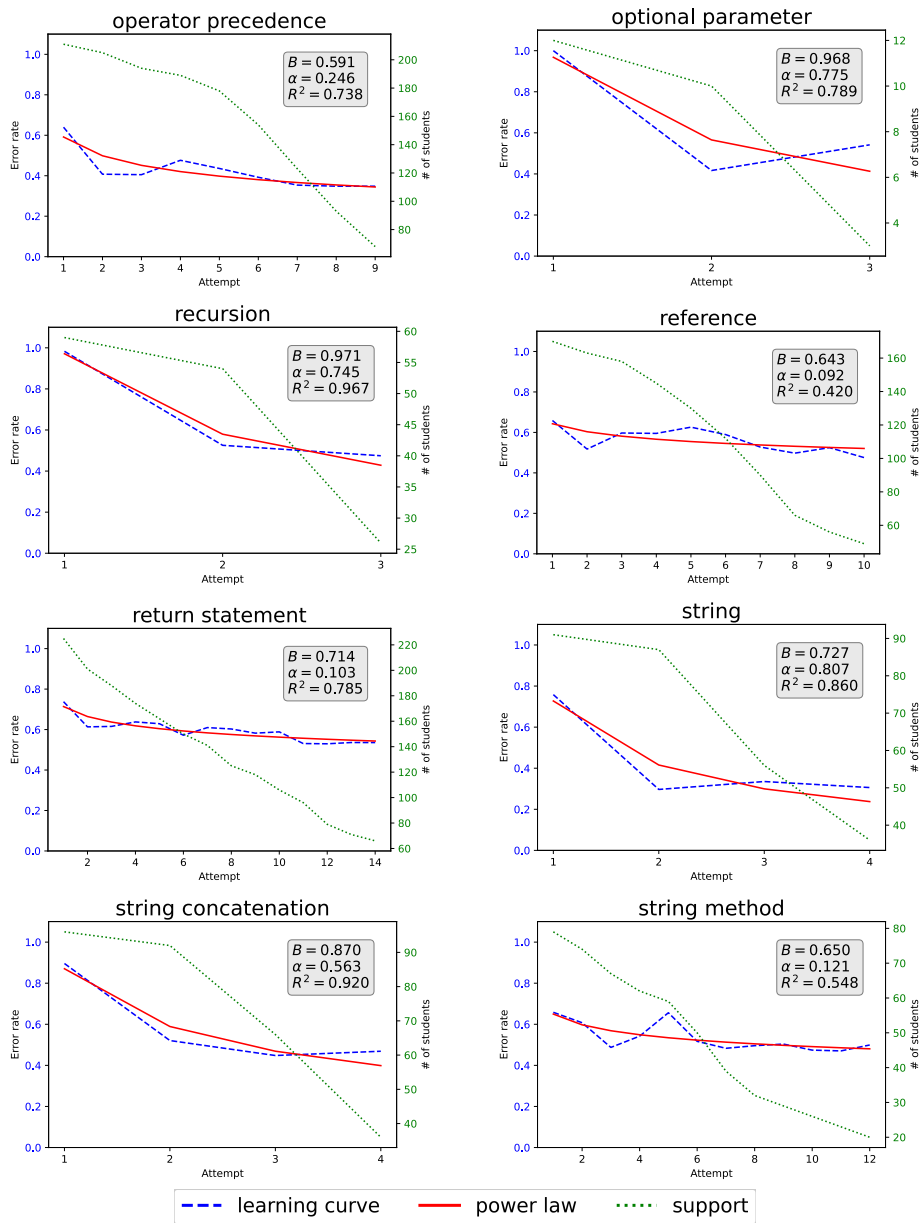


Figure A.5: Learning curves (5/6).

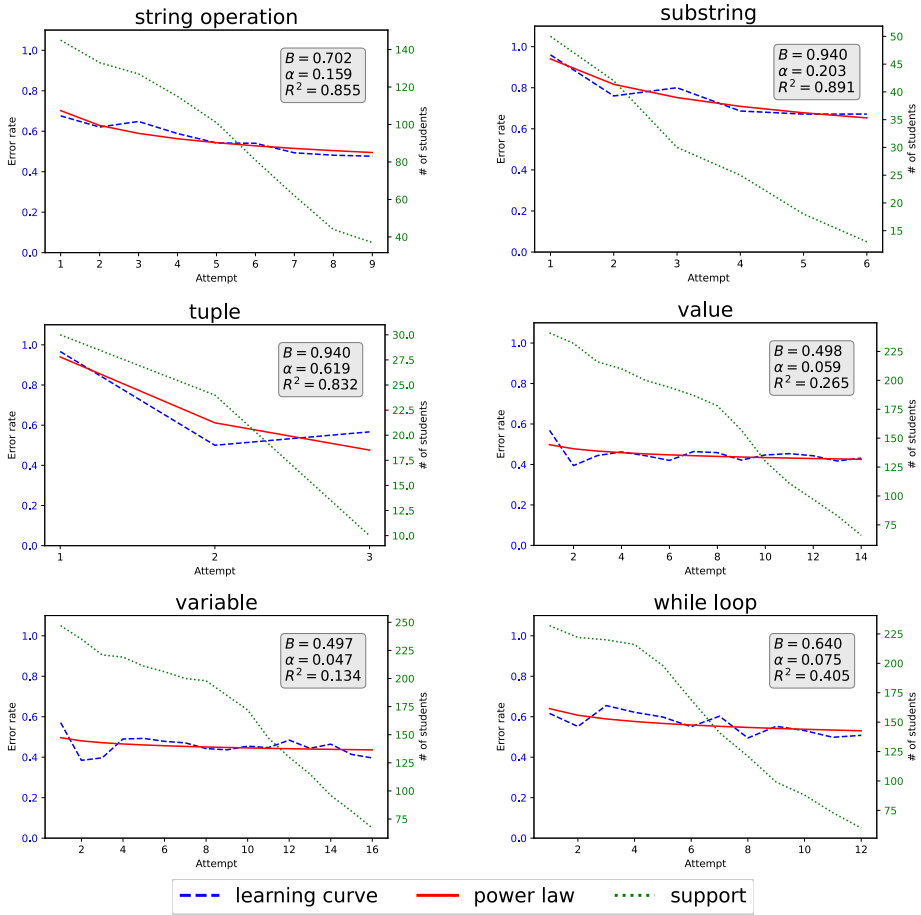


Figure A.6: Learning curves (6/6).

APPENDIX B

Implementation

Intelligent Textbooks or *InTextbooks* is the name for the collection of tools developed using this thesis' approach. So far, three implementations have been created:

- *ITCore* is the main library capable of extracting the knowledge models from textbooks. All stages of the approach have been implemented using Java. Some tasks required services developed in Python (e.g., term recognition in the text). An open-source version of the library is available as a GitHub project at <https://github.com/intextbooks/ITCore>.
- *ITService* is a public web service based on *ITCore*. Users can upload a textbook, then the server will process the textbook and create the knowledge model. Users receive an email with a link to download the model when it is ready. The services is available at <https://intextbooks.science.uu.nl/>.
- *InTextbooks System* is a platform where students can engage and interact with textbooks in several ways. As discussed in Chapter 7, the system is in the stage of design with a basic functional prototype.

The <https://intextbooks.science.uu.nl/> site can be consulted for an up-to-date list of the tools developed using the approach presented in this thesis.

Bibliography

- [1] Abiteboul, S., Buneman, P. & Suciu, D., *Data on the web: from relations to semistructured data and XML*, Morgan Kaufmann, 2000.
- [2] ADL Initiative, *Shareable Content Object Reference Model. Technical Specification (4th Ed.)* https://www.adlnet.gov/assets/uploads/SCORM_2004_4ED_v1_1_Doc_Suite.zip, 2004.
- [3] ADL Initiative, *Experience API*, <https://github.com/adlnet/xAPI-Spec/blob/master/xAPI-About.md>, 2017.
- [4] Adorni, G., Alzetta, C., Kocova, F., Passalacqua, S. & Torre, I., “Towards the identification of propaedeutic relations in textbooks”, in: *International Conference on Artificial Intelligence in Education*, Springer, 2019, pp. 1–13.
- [5] Aghaebrahimian, A. & Cieliebak, M., “Named entity disambiguation at scale”, in: *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, Springer, 2020, pp. 102–110.
- [6] Agirre, E. & Rigau, G., “Word Sense Disambiguation using Conceptual Density”, *Proceedings of the 16th conference on Computational linguistics -*, vol. 1, 1996, p. 8.
- [7] Agrawal, R., Gollapudi, S., Kannan, A. & Kenthapadi, K., “Study navigator: an algorithmically generated aid for learning from electronic textbooks”, *Journal of Educational Data Mining*, vol. 6, no. 1, 2014.
- [8] Alani, H., Kim, S., Millard, D. E., Weal, M. J., Hall, W., Lewis, P. H. & Shadbolt, N. R., “Automatic ontology-based knowledge extraction from web documents”, *IEEE Intelligent Systems*, vol. 18, no. 1, 2003, pp. 14–21.
- [9] Alpizar-Chacon, I., Barria-Pineda, J., Akhuseyinoglu, K., Sosnovsky, S. & Brusilovsky, P., “Integrating textbooks with smart interactive content for learning programming”, in: *Proceedings of the Third Workshop on Intelligent Textbooks*, vol. 2895, CEUR WS, 2021, pp. 4–18.
- [10] Alpizar-Chacon, I., Hart, M. van der, Wiersma, Z. S., Theunissen, L. S. & Sosnovsky, S., “Transformation of pdf textbooks into intelligent educational resources”, in: *Proceedings of the Second Workshop on Intelligent Textbooks*, vol. 2674, CEUR-WS, 2020, pp. 4–16.
- [11] Alpizar-Chacon, I. & Sosnovsky, S., “Expanding the web of knowledge: one textbook at a time”, in: *Proceedings of the 30th on Hypertext and Social Media, HT '19*, Hof, Germany: ACM, 2019.

- [12] Alpizar-Chacon, I. & Sosnovsky, S., “Interlingua: linking textbooks across different languages”, in: *Proceedings of the First Workshop on Intelligent Textbooks*, vol. 2384, CEUR-WS, 2019, pp. 104–117.
- [13] Alpizar-Chacon, I. & Sosnovsky, S., “Order out of chaos: construction of knowledge models from pdf textbooks”, in: *Proceedings of the ACM Symposium on Document Engineering 2020*, 2020, pp. 1–10.
- [14] Alpizar-Chacon, I. & Sosnovsky, S., “Knowledge models from pdf textbooks”, *New Review of Hypermedia and Multimedia*, vol. 27, no. 1-2, 2021, pp. 128–176.
- [15] Alpizar-Chacon, I. & Sosnovsky, S., “What’s in an index: extracting domain-specific knowledge graphs from textbooks”, in: *Proceedings of the ACM Web Conference 2022 (WWW ’22)*, 2022, pp. 966–976.
- [16] Alpizar-Chacon, I. & Sosnovsky, S., “Measuring the quality of domain models extracted from textbooks with learning curves analysis”, *In submission*, n.d.
- [17] Ament, K., *Indexing: a nuts-and-bolts guide for technical writers*, William Andrew, 2001.
- [18] Anderson, J. R., Corbett, A. T., Koedinger, K. R. & Pelletier, R., “Cognitive tutors: lessons learned”, *The journal of the learning sciences*, vol. 4, no. 2, 1995, pp. 167–207.
- [19] Anderson, J. & Reiser, B., “The lisp tutor”, *Byte*, vol. 10, no. 4, 1985, pp. 159–175.
- [20] Arasu, A. & Garcia-Molina, H., “Extracting structured data from web pages”, in: *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 2003, pp. 337–348.
- [21] Asaishi, T., “An analysis of the terminological structure of index terms in textbooks”, *Procedia-Social and Behavioral Sciences*, vol. 27, 2011, pp. 209–217.
- [22] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R. & Ives, Z., “Dbpedia: a nucleus for a web of open data”, in: *The semantic web*, Springer, 2007, pp. 722–735.
- [23] Baker, J. B., Sexton, A. P. & Sorge, V., “A linear grammar approach to mathematical formula recognition from pdf”, in: *International Conference on Intelligent Computer Mathematics*, Springer, 2009, pp. 201–216.
- [24] Baker-Eveleth, L. & Stone, R. W., “Usability, expectation, confirmation, and continuance intentions to use electronic textbooks”, *Behaviour & Information Technology*, vol. 34, no. 10, 2015, pp. 992–1004.
- [25] Barria-Pineda, J., Brusilovsky, P. & He, D., “Reading mirror: social navigation and social comparison for electronic textbooks”, in: *Proceedings of the First Workshop on Intelligent Textbooks*, vol. 2384, CEUR-WS, 2019, pp. 104–117.
- [26] Bast, H. & Korzen, C., “A benchmark and evaluation for text extraction from pdf”, in: *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, IEEE, 2017, pp. 1–10.
- [27] Bayomi, M. & Lawless, S., “C-hits: a concept-based hierarchical text segmentation approach”, in: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, 2018.

- [28] Bender, M. A., Farach-Colton, M., Pemmasani, G., Skiena, S. & Sumazin, P., "Lowest common ancestors in trees and directed acyclic graphs", *Journal of Algorithms*, vol. 57, no. 2, 2005, pp. 75–94.
- [29] *Best Practices for Indexing*, 1st ed., American Society for Indexing (ASI), 2015.
- [30] Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R. & Hellmann, S., "Dbpedia-a crystallization point for the web of data", *Web Semantics: science, services and agents on the world wide web*, vol. 7, no. 3, 2009, pp. 154–165.
- [31] Blei, D. M., Ng, A. Y. & Jordan, M. I., "Latent dirichlet allocation", *Journal of machine Learning research*, vol. 3, no. Jan, 2003, pp. 993–1022.
- [32] Blumberg, R. & Atre, S., "The problem with unstructured data", *Dm Review*, vol. 13, no. 42-49, 2003, p. 62.
- [33] Bologna Declaration, *Towards the European Higher Education Area*, <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=LEGISUM:c11088>, Bologna, Italy, June 1999.
- [34] Booth, P. F., *Indexing The Manual of Good Practice*, De Gruyter, 2013.
- [35] Bouarroudj, W. & Boufaïda, Z., "A candidate generation algorithm for named entities disambiguation using dbpedia", in: *Trends and Advances in Information Systems and Technologies: Volume 1 6*, Springer, 2018, pp. 712–721.
- [36] Boyle, T., Gray, G., Wendl, B & Davies, M., "Taking the plunge with clem: the design and evaluation of a large scale cal system", *Computers and Education*, vol. 22, no. 1/2, 1994, pp. 19–26.
- [37] Brown, M. H. & Najork, M. A., "Collaborative active textbooks", *Journal of Visual Languages and Computing*, vol. 8, no. 4, 1997, pp. 453–486.
- [38] Brusilovsky, P., "Developing adaptive educational hypermedia systems: from design models to authoring tools", *Authoring Tools for Advanced Technology Learning Environments: Toward Cost-Effective Adaptive, Interactive and Intelligent Educational Software*, 2003, pp. 377–409.
- [39] Brusilovsky, P., "Knowledge tree: a distributed architecture for adaptive e-learning", in: *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, 2004, pp. 104–113.
- [40] Brusilovsky, P., "Adaptive navigation support", in: *The adaptive web*, Springer, 2007, pp. 263–290.
- [41] Brusilovsky, P., Edwards, S., Kumar, A., Malmi, L., Benotti, L., Buck, D., Ihan-tola, P., Prince, R., Sirkiä, T., Sosnovsky, S., et al., "Increasing adoption of smart learning content for computer science education", in: *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference*, 2014, pp. 31–57.
- [42] Brusilovsky, P. & Eklund, J., "A study of user model based link annotation in educational hypermedia", *Journal of Universal Computer Science*, vol. 4, no. 4, 1998, pp. 429–448.
- [43] Brusilovsky, P. & Peylo, C., "Adaptive and intelligent web-based educational systems", *International Journal of Artificial Intelligence in Education*, vol. 13, no. 2–4, Apr. 2003, 159–172.

- [44] Brusilovsky, P., Schwarz, E. & Weber, G., “ELM-ART: an intelligent tutoring system on world wide web”, in: *Third International Conference on Intelligent Tutoring Systems, ITS-96*, ed. by Frasson, C., Gauthier, G. & Lesgold, A., vol. 1086, Lecture Notes in Computer Science, Springer Verlag, 1996, pp. 261–269.
- [45] Brusilovsky, P., Schwarz, E. & Weber, G., “Electronic textbooks on www: from static hypertext to interactivity and adaptivity”, in: *Web Based Instruction*, ed. by Khan, B. H., Englewood Cliffs, New Jersey: Educational Technology Publications, 1997, pp. 255–261.
- [46] Brusilovsky, P., Sosnovsky, S. & Shcherbinina, O., “User modeling in a distributed e-learning architecture”, in: *International conference on user modeling*, Springer, 2005, pp. 387–391.
- [47] Burton-Jones, A., Storey, V. C., Sugumaran, V. & Ahluwalia, P., “A semiotic metrics suite for assessing the quality of ontologies”, *Data & Knowledge Engineering*, vol. 55, no. 1, 2005, pp. 84–102.
- [48] Caenepeel, S. & Groen, P. de, *Waarschijnlijkheidsrekening en statistiek*, Vrije Universiteit Brussel, 2002.
- [49] Chambliss, M. J., “The characteristics of well-designed science textbooks”, *The psychology of science text comprehension*, 2002, pp. 51–72.
- [50] Chambliss, M. J. & Calfee, R. C., “Designing science textbooks to enhance student understanding”, *Educational psychologist*, vol. 24, no. 3, 1989, pp. 307–322.
- [51] Chang, A. X., Spitkovsky, V. I., Manning, C. D. & Agirre, E., “A comparison of Named-Entity Disambiguation and Word Sense Disambiguation”, *Lrec*, 2016, pp. 860–867.
- [52] Chao, H. & Fan, J., “Layout and content extraction for pdf documents”, in: *International Workshop on Document Analysis Systems*, Springer, 2004, pp. 213–224.
- [53] Chaplot, D. S., Yang, Y., Carbonell, J. & Koedinger, K. R., “Data-driven automated induction of prerequisite structure graphs.”, *International Educational Data Mining Society*, 2016.
- [54] Chau, H., Labutov, I., Thaker, K., He, D. & Brusilovsky, P., “Automatic concept extraction for domain and student modeling in adaptive textbooks”, *International Journal of Artificial Intelligence in Education*, vol. 31, no. 4, 2021, pp. 820–846.
- [55] Chaudhri, V. K., Boggess, M., Aung, H. L., Mallick, D. B., Waters, A. C. & Baraniuk, R., “A case study in bootstrapping ontology graphs from textbooks”, in: *3rd Conference on Automated Knowledge Base Construction*, 2021.
- [56] Chaudhri, V. K., Cheng, B., Overholtzer, A., Roschelle, J., Spaulding, A., Clark, P., Greaves, M. & Gunning, D., “Inquire biology: a textbook that answers questions”, *AI Magazine*, vol. 34, no. 3, 2013, pp. 55–72.
- [57] Chiappetta, E. L., Fillman, D. A. & Sethna, G. H., “A method to quantify major themes of scientific literacy in science textbooks”, *Journal of research in science teaching*, vol. 28, no. 8, 1991, pp. 713–725.

- [58] Chiu, T. K. F., “Introducing electronic textbooks as daily-use technology in schools: a top-down adoption process”, *British Journal of Educational Technology*, vol. 48, no. 2, 2017, pp. 524–537.
- [59] Clark, C. & Divvala, S., “Pdffigures 2.0: mining figures from research papers”, in: *2016 IEEE/ACM Joint Conference on Digital Libraries (JCDL)*, IEEE, 2016, pp. 143–152.
- [60] Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C., *Introduction to Algorithms, Third Edition*, 3rd, The MIT Press, 2009, ISBN: 0262033844.
- [61] Councill, I. G., Giles, C. L. & Kan, M. yen, “Parscit: an open-source crf reference string parsing package”, in: *International Language Resources and Evaluation*, European Language Resources Association, 2008.
- [62] Csomai, A. & Mihalcea, R., “Investigations in unsupervised back-of-the-book indexing.”, in: *FLAIRS Conference*, 2007, pp. 211–216.
- [63] Daiber, J., Jakob, M., Hokamp, C. & Mendes, P. N., “Improving efficiency and accuracy in multilingual entity extraction”, in: *Proceedings of the 9th international conference on semantic systems*, 2013, pp. 121–124.
- [64] Dalgaard, P., *Introductory statistics with R*, Lightning Source UK Ltd., 2011.
- [65] De Bra, P. M., “Teaching through adaptive hypertext on the www”, *International Journal of Educational Telecommunications*, vol. 3, no. 2, 1997, pp. 163–179.
- [66] De Vocht, L., Coppens, S., Verborgh, R., Vander Sande, M., Mannens, E. & Walle, R. Van de, “Discovering meaningful connections between resources in the web of data”, in: *LDOW*, 2013.
- [67] De Vries, H., Elliott, M. N., Kanouse, D. E. & Teleki, S. S., “Using pooled kappa to summarize interrater agreement across many items”, *Field methods*, vol. 20, no. 3, 2008, pp. 272–282.
- [68] Déjean, H. & Meunier, J.-L., “On tables of contents and how to recognize them”, *International Journal of Document Analysis and Recognition (IJDAR)*, vol. 12, no. 1, 2009, pp. 1–20.
- [69] Dekking, F. M., Kraaikamp, C., Lopuhaä, H. P. & Meester, L. E., *A modern introduction to probability and statistics: understanding why and how*, Springer, 2005.
- [70] Demartini, G., Difallah, D. E. & Cudré-Mauroux, P., “Large-scale linked data integration using probabilistic reasoning and crowdsourcing”, *The VLDB Journal*, vol. 22, no. 5, Oct. 2013, pp. 665–687.
- [71] DeNoyelles, A. & Raible, J., “Exploring the use of e-textbooks in higher education: a multiyear study”, *Educause Review*, 2017.
- [72] Devore, J. L. & Berk, K. N., *Modern Mathematical Statistics with Applications*, Springer, 2012.
- [73] Di Noia, T., Ostuni, V. C., Rosati, J., Tomeo, P., Di Sciascio, E., Mirizzi, R. & Bartolini, C., “Building a relatedness graph from linked open data: a case study in the it domain”, *Expert Systems with Applications*, vol. 44, 2016, pp. 354–366.
- [74] Di Vesta, F. J. & Gray, G. S., “Listening and note taking.”, *Journal of educational psychology*, vol. 63, no. 1, 1972, p. 8.

- [75] 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)*, IEEE, 2006, pp. 919–925.
- [76] Diez, D. M., Cetinkaya-Rundel, M. & Barr, C. D., *OpenIntro statistics*, Fourth, openintro.org, 2019.
- [77] Dolog, P. & Nejdl, W., “Semantic web technologies for the adaptive web”, in: *The adaptive web*, Springer, 2007, pp. 697–719.
- [78] Doucet, A., Kazai, G., Colutto, S. & Mühlberger, G., “Icdar 2013 competition on book structure extraction”, in: *12th International Conference on Document Analysis and Recognition*, IEEE, 2013, pp. 1438–1443.
- [79] Doucet, A., Kazai, G., Dresevic, B., Uzelac, A., Radakovic, B. & Todoc, N., “Setting up a competition framework for the evaluation of structure extraction from ocr-ed books”, *International Journal on Document Analysis and Recognition (IJ DAR)*, vol. 14, 2011, pp. 45–52.
- [80] Downey, A., *Think Python*, 2nd, Green Tea Press, 2015.
- [81] Dresscher, L., Alpizar-Chacon, I. & Sosnovsky, S., “Generation of assessment questions from textbooks enriched with knowledge models”, in: *Proceedings of the Third Workshop on Intelligent Textbooks*, vol. 2895, CEUR-WS, 2021, pp. 45–59.
- [82] Dwarakanath, A., Ramnani, R. R. & Sengupta, S., “Automatic extraction of glossary terms from natural language requirements”, in: *2013 21st IEEE International Requirements Engineering Conference (RE)*, IEEE, 2013, pp. 314–319.
- [83] Easton, V. J. & McColl, J. H., *Statistics Glossary from STEPS*, <http://www.stats.gla.ac.uk/steps/glossary/index.html>, [Online; accessed 12-2020], 1997.
- [84] Eckstein, P. P., *Repetitorium Statistik*, Springer Gabler, 2013.
- [85] Edwards, S. H. & Murali, K. P., “Codeworkout: short programming exercises with built-in data collection”, in: *2017 Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE'17*, ACM Press, 2014, pp. 188–193.
- [86] Eikvil, L., “Ocr-optical character recognition”, *Norsk Regnesentral*, vol. 876, 1993.
- [87] Engbrecht, J. R., “Digital textbooks versus print textbooks”, *Culminating Projects in Teacher Development*, 2018.
- [88] Ericson, B., “An analysis of interactive feature use in two ebooks”, in: *Proceedings of the First Workshop on Intelligent Textbooks*, vol. 2384, CEUR-WS, 2019, pp. 4–17.
- [89] European Commission, *Youth on the Move - an initiative to unleash the potential of young people to achieve smart, sustainable and inclusive growth in the European Union*, http://europa.eu/youthonthemove/docs/communication/youth-on-the-move_EN.pdf, 2010.
- [90] Faber, M. H., *Statistics and probability theory: in pursuit of engineering decision support*, Springer Verlag, 2012.
- [91] Fahrmeir, L., *Statistik: der Weg zur Datenanalyse*, Springer, 2007.

- [92] Fan, J., Kalyanpur, A., Gondek, D. C. & Ferrucci, D. A., "Automatic knowledge extraction from documents", *IBM Journal of Research and Development*, vol. 56, no. 3.4, 2012, pp. 5–1.
- [93] Fang, J., Gao, L., Bai, K., Qiu, R., Tao, X. & Tang, Z., "A table detection method for multipage pdf documents via visual separators and tabular structures", in: *2011 International Conference on Document Analysis and Recognition*, IEEE, 2011, pp. 779–783.
- [94] Färber, M., Ell, B., Menne, C. & Rettinger, A., "A comparative survey of dbpedia, freebase, opencyc, wikidata, and yago", *Semantic Web Journal*, vol. 1, no. 1, 2015, pp. 1–5.
- [95] Fayyad, U., Piatetsky-Shapiro, G. & Smyth, P., "The kdd process for extracting useful knowledge from volumes of data", *Communications of the ACM*, vol. 39, no. 11, 1996, pp. 27–34.
- [96] Feldstein, A. P., Martin, M., Hudson, A., Warren, K., Hilton III, J. & Wiley, D., "Open textbooks and increased student access and outcomes", *European Journal of Open, Distance and E-Learning*, 2012.
- [97] Ferragina, P. & Scaiella, U., "Fast and Accurate Annotation of Short Texts with Wikipedia Pages", *IEEE Software*, vol. 29, no. 1, 2012, pp. 70–75, arXiv: 1006.3498.
- [98] Ferreira-Mello, R., André, M., Pinheiro, A., Costa, E. & Romero, C., "Text mining in education", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 9, no. 6, 2019, e1332.
- [99] Fiallos, A. & Ochoa, X., "Semi-automatic generation of intelligent curricula to facilitate learning analytics", in: *Proceedings of the 9th International Conference on Learning Analytics & Knowledge*, 2019, pp. 46–50.
- [100] Filler, M. G. & DiGabriele, J. A., *A Quantitative Approach to Commercial Damages: Applying Statistics to the Measurement of Lost Profits*, John Wiley & Sons, 2012.
- [101] Finkelstein, M. O., *Basic concepts of probability and statistics in the law*, Springer, 2009.
- [102] Fisher, J. L. & Harris, M. B., "Effect of note taking and review on recall.", *Journal of Educational Psychology*, vol. 65, no. 3, 1973, p. 321.
- [103] Ford, K. M. & Adams-Webber, J. R., "Knowledge acquisition and constructivist epistemology", in: *The psychology of expertise*, Springer, 1992, pp. 121–136.
- [104] Gaines, B. R., "Knowledge acquisition: past, present and future", *Int. J. Hum.-Comput. Stud.*, vol. 71, no. 2, Feb. 2013, 135–156.
- [105] Gaizauskas, R., Barker, E., Paramita, M. L. & Aker, A., *Assigning Terms to Domains by Document Classification*, tech. rep., 2014, pp. 11–21, URL: <https://demo.taas-project.eu/domains..>
- [106] Gaizauskas, R. & Wilks, Y., "Information extraction: beyond document retrieval", *Journal of documentation*, 1998.
- [107] Gall, M. D., "The use of questions in teaching", *Review of educational research*, vol. 40, no. 5, 1970, pp. 707–721.

- [108] Gao, L., Tang, Z., Lin, X., Liu, Y., Qiu, R. & Wang, Y., “Structure extraction from pdf-based book documents”, in: *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*, 2011, pp. 11–20.
- [109] Gao, L., Tang, Z., Lin, X. & Qiu, R., “Comprehensive global typography extraction system for electronic book documents”, in: *2008 The Eighth IAPR International Workshop on Document Analysis Systems*, IEEE, 2008, pp. 615–621.
- [110] Gao, L., Tang, Z., Lin, X., Tao, X. & Chu, Y., “Analysis of book documents’ table of content based on clustering”, in: *2009 10th International Conference on Document Analysis and Recognition*, IEEE, 2009, pp. 911–915.
- [111] Giannini, S., Colucci, S., Donini, F. M. & Di Sciascio, E., “A logic-based approach to named-entity disambiguation in the web of data”, in: *IA 2015 Advances in Artificial Intelligence: XIVth International Conference of the Italian Association for Artificial Intelligence*, Springer, 2015, pp. 367–380.
- [112] Glushko, R. J., “The discipline of organizing”, *BULLETIN of the American society for information science and technology*, vol. 40, no. 1, 2013, pp. 21–27.
- [113] Gómez-Pérez, A., “Towards a framework to verify knowledge sharing technology”, *Expert Systems with applications*, vol. 11, no. 4, 1996, pp. 519–529.
- [114] Govindan, V. & Shivaprasad, A., “Character recognition—a review”, *Pattern recognition*, vol. 23, no. 7, 1990, pp. 671–683.
- [115] Gruber, T. R., “Toward principles for the design of ontologies used for knowledge sharing?”, *International journal of human-computer studies*, vol. 43, no. 5-6, 1995, pp. 907–928.
- [116] Guerra, J., Hosseini, R., Somyurek, S. & Brusilovsky, P., “An intelligent interface for learning content: combining an open learner model and social comparison to support self-regulated learning and engagement”, in: *Proceedings of the 21st international conference on intelligent user interfaces*, 2016, pp. 152–163.
- [117] Guerra, J., Sosnovsky, S. & Brusilovsky, P., “When one textbook is not enough: linking multiple textbooks using probabilistic topic models”, in: *European conference on technology enhanced learning*, Springer, 2013, pp. 125–138.
- [118] Guerra Hollstein, J. D., “Open learner models for self-regulated learning: exploring the effects of social comparison and granularity”, PhD thesis, University of Pittsburgh, 2018.
- [119] Gunetti, D. & Picardi, C., “Keystroke analysis of free text”, *ACM Transactions on Information and System Security (TISSEC)*, vol. 8, no. 3, 2005, pp. 312–347.
- [120] Gutttag, J., *Introduction to computation and programming using Python, Revised and Expanded Edition*, The MIT Press, 2013.
- [121] Hahm, Y., Park, J., Lim, K., Kim, Y., Hwang, D. & Choi, K.-S., “Named Entity Corpus Construction using Wikipedia and DBpedia Ontology”, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*, 2014, pp. 2565–2569.

- [122] Hall, K. M., Sabey, B. L. & McClellan, M., “Expository text comprehension: helping primary-grade teachers use expository texts to full advantage”, *Reading psychology*, vol. 26, no. 3, 2005, pp. 211–234.
- [123] Han, X. & Sun, L., “A generative entity-mention model for linking entities with knowledge base”, in: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, Association for Computational Linguistics, 2011, pp. 945–954.
- [124] Han, X., Liu, Z. & Sun, M., “Neural knowledge acquisition via mutual attention between knowledge graph and text”, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 1, 2018.
- [125] Hardt, D., *The OAuth 2.0 authorization framework*, Standard, Internet Engineering Task Force (IETF), 2012.
- [126] Hartig, O. & Freytag, J.-C., “Foundations of traversal based query execution over linked data”, in: *Proceedings of the 23rd ACM conference on Hypertext and social media*, 2012, pp. 43–52.
- [127] Hassan, T., “Object-level document analysis of pdf files”, in: *Proceedings of the 9th ACM symposium on Document engineering*, 2009, pp. 47–55.
- [128] Herder, E., Sosnovsky, S. & Dimitrova, V., “Adaptive intelligent learning environments”, in: *Technology Enhanced Learning*, Springer, 2017, pp. 109–114.
- [129] Hjørland, B., “What is knowledge organization (ko)?”, *KO Knowledge Organization*, vol. 35, no. 2-3, 2008, pp. 86–101.
- [130] Hoffart, J., Seufert, S., Nguyen, D. B., Theobald, M. & Weikum, G., “Kore: keyphrase overlap relatedness for entity disambiguation”, in: *Proceedings of the 21st ACM international conference on Information and knowledge management*, ACM, 2012, pp. 545–554.
- [131] Hosseini, R. & Brusilovsky, P., “Javaparser: a fine-grain concept indexing tool for java problems”, in: *The First Workshop on AI-supported Education for Computer Science (AIEDCS 2013)*, 2013, pp. 60–63.
- [132] Hughes, J., Cantor, S., Hodges, J., Hirsch, F., Mishra, P., Philpott, R. & Maler, E., “Profiles for the oasis security assertion markup language (saml) v2. 0”, *OASIS standard*, 2005.
- [133] Hulpus, I., Prangnawarat, N. & Hayes, C., “Path-Based Semantic Relatedness on Linked Data and Its Use to Word and Entity Disambiguation”, in: *Proceedings of the 14th International Semantic Web Conference (ISWC 2015)*, vol. 9366, 2015, pp. 442–457.
- [134] IEEE, *IEEE Standard for Learning Object Metadata*, https://standards.ieee.org/standard/1484_12_1-2020.html, 2020.
- [135] Illowsky, B. & Dean, S. L., *Introductory statistics*, OpenStax, Rice University, 2018.
- [136] IMS Global, *Learning Tools Interoperability Core Specification*, <https://www.imsglobal.org/spec/lti/latest/>, 2019.
- [137] IMS Global, *Caliper Analytics Specification*, <https://www.imsglobal.org/spec/caliper/latest/>, 2020.
- [138] International Statistical Institute, *ISI Multilingual Glossary of Statistical Terms*, <http://isi.cbs.nl/glossary/>, [Online; accessed 12-2020], 2011.

- [139] Järvelin, K. & Kekäläinen, J., “Cumulated gain-based evaluation of ir techniques”, *ACM Transactions on Information Systems*, vol. 20, no. 4, 2002, pp. 422–446.
- [140] Javadian Sabet, A., Alpizar-Chacon, I., Barria-Pineda, J., Brusilovsky, P. & Sosnovsky, S., “Enriching intelligent textbooks with interactivity: when smart content allocation goes wrong”, in: *Proceedings of the Fourth Workshop on Intelligent Textbooks*, vol. 3192, CEUR-WS, 2022, pp. 92–104.
- [141] Jones, P., *Knowledge acquisition*, The AGRIS database, 1989.
- [142] Jourdain, B., *Probabilités et statistique*, 2013.
- [143] Kaiya, H. & Saeki, M., “Using domain ontology as domain knowledge for requirements elicitation”, in: *14th IEEE International Requirements Engineering Conference (RE’06)*, IEEE, 2006, pp. 189–198.
- [144] Kaltenbach, H.-M., *A concise guide to statistics*, Springer, 2012.
- [145] Kasdorf, W. E., *The Columbia guide to digital publishing*, Columbia University Press, 2003.
- [146] Kavcic, A., “Fuzzy user modeling for adaptation in educational hypermedia”, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 34, no. 4, 2004, pp. 439–449.
- [147] Kawamata, T., Matsuda, Y., Sekiya, T. & Yamaguchi, K., “Analysis of computer science textbooks by topic modeling and dynamic time warping”, in: *2021 IEEE International Conference on Engineering, Technology & Education (TALE)*, IEEE, 2021, pp. 865–870.
- [148] Kay, J., Kummerfeld, B. & Lauder, P., “Personis: a server for user models”, in: *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, Springer, 2002, pp. 203–212.
- [149] Kazil, J. & Jarmul, K., *Data wrangling with Python*, O’Reilly, 2016.
- [150] Kenny, A., *Ancient Philosophy: A New History of Western Philosophy, Volume 1*, OUP Oxford, 2004.
- [151] Kern, R., Jack, K., Hristakeva, M. & Granitzer, M., “Teambeam-meta-data extraction from scientific literature”, *D-Lib Magazine*, vol. 18, no. 7, 2012, p. 1.
- [152] Kern, R. & Klampfl, S., “Extraction of references using layout and formatting information from scientific articles”, *D-Lib Magazine*, vol. 19, no. 9-10, 2013.
- [153] Kida, M., Tonoike, M., Utsuro, T. & Sato, S., “Domain classification of technical terms using the web”, *Systems and Computers in Japan*, vol. 38, no. 14, 2007, pp. 11–19.
- [154] Kissinger, J. S., “The social & mobile learning experiences of students using mobile e-books.”, *Journal of Asynchronous Learning Networks*, vol. 17, no. 1, 2013, pp. 155–170.
- [155] Kobilarov, G., Scott, T., Raimond, Y., Oliver, S., Sizemore, C., Smethurst, M., Bizer, C. & Lee, R., “Media meets semantic web—how the bbc uses dbpedia and linked data to make connections”, in: *The Semantic Web: Research and Applications: 6th European Semantic Web Conference*, Springer, 2009, pp. 723–737.

- [156] Kodratoff, Y., “Knowledge discovery in texts: a definition, and applications”, in: *International Symposium on Methodologies for Intelligent Systems*, Springer, 1999, pp. 16–29.
- [157] Koedinger, K. R., Baker, R. S., Cunningham, K., Skogsholm, A., Leber, B. & Stamper, J., “A data repository for the edm community: the pslc datashop”, *Handbook of educational data mining*, vol. 43, 2010, pp. 43–56.
- [158] Koedinger, K. R., Corbett, A. T. & Perfetti, C., “The knowledge-learning-instruction framework: bridging the science-practice chasm to enhance robust student learning”, *Cognitive science*, vol. 36, no. 5, 2012, pp. 757–798.
- [159] Kruit, B., He, H. & Urbani, J., “Tab2know: building a knowledge base from tables in scientific papers”, in: *International Semantic Web Conference*, Springer, 2020, pp. 349–365.
- [160] Kuhn, D., Garcia-Mila, M., Zohar, A., Andersen, C., White, S. H., Klahr, D. & Carver, S. M., “Strategies of knowledge acquisition”, *Monographs of the society for research in child development*, 1995, pp. i–157.
- [161] Labutov, I., Huang, Y., Brusilovsky, P. & He, D., “Semi-supervised techniques for mining learning outcomes and prerequisites”, in: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 907–915.
- [162] Lalithsena, S., Perera, S., Kapanipathi, P. & Sheth, A., “Domain-specific hierarchical subgraph extraction: a recommendation use case”, in: *2017 IEEE International Conference on Big Data (Big Data)*, IEEE, 2017, pp. 666–675.
- [163] Landrum, R. E., “Core terms in undergraduate statistics”, *Teaching of Psychology*, vol. 32, no. 4, 2005.
- [164] Larrañaga, M., Conde, A., Calvo, I., Elorriaga, J. A. & Arruarte, A., “Automatic generation of the domain module from electronic textbooks: method and validation”, *IEEE Trans. on Knowl. and Data Eng.*, vol. 26, no. 1, Jan. 2014, pp. 69–82.
- [165] Larrañaga, M., Rueda, U., Elorriaga, J. A. & Arruarte, A., “Acquisition of the domain structure from document indexes using heuristic reasoning”, in: *International Conference on Intelligent Tutoring Systems*, Springer, 2004, pp. 175–186.
- [166] Leal, J. P., Rodrigues, V. & Queirós, R., “Computing semantic relatedness using dbpedia”, in: *Symposium on Languages, Applications and Technologies*, 1st, Schloss Dagstuhl, 2012, pp. 133–147.
- [167] Liesaputra, V. & Witten, I. H., “Seeking information in realistic books: a user study”, in: *Joint Conference on Digital Libraries, JCDL 2008*, 2008, pp. 29–38.
- [168] Lin, X., Gao, L., Tang, Z., Baker, J. & Sorge, V., “Mathematical formula identification and performance evaluation in pdf documents”, *International Journal on Document Analysis and Recognition (IJ DAR)*, vol. 17, no. 3, 2014, pp. 239–255.
- [169] Lin, X., Gao, L., Tang, Z., Lin, X. & Hu, X., “Mathematical formula identification in pdf documents”, in: *2011 international conference on document analysis and recognition*, IEEE, 2011, pp. 1419–1423.
- [170] Linn, M., “How can hypermedia tools help teach programming”, *Learning and Instruction*, vol. 2, 1992, pp. 119–139.

- [171] Lopes, L., Vieira, R., Finatto, M. J. & Martins, D., “Extracting compound terms from domain corpora”, *Journal of the Brazilian Computer Society*, vol. 16, no. 4, Nov. 2010, pp. 247–259.
- [172] Lopes, L., Vieira, R., Finatto, M. J. B., Zanette, A., Martins, D. & Ribeiro Junior, L. C., “Automatic extraction of composite terms for construction of ontologies: an experiment in the health care area”, *RECIIS: electronic journal of communication & innovation in health. Rio de Janeiro, RJ*, 2009.
- [173] Lovegrove, W. S. & Brailsford, D. F., “Document analysis of pdf files: methods, results and implications”, *Electronic Publishing–Origination, Dissemination and Design*, vol. 8, no. 3, 1995.
- [174] Madsen, B., *Statistics for Non-Statisticians*, Springer, 2011.
- [175] Mahmood, K., “Conformity to quality characteristics of textbooks: the illusion of textbook evaluation in pakistan”, *Journal of research and Reflections in Education*, vol. 5, no. 2, 2011, pp. 170–190.
- [176] Mandl, H. & Levin, J. R., *Knowledge acquisition from text and pictures*, Elsevier, 1989.
- [177] Manning, C. D., Raghavan, P. & Schütze, H., *An Introduction to Information Retrieval*, Cambridge University Press, 2009.
- [178] Marchant, T., *Statistiek I*, Universiteit Gent, 2012.
- [179] Marie, N., Gandon, F., Ribière, M. & Rodio, F., “Discovery hub: on-the-fly linked data exploratory search”, in: *Proceedings of the 9th international conference on semantic systems*, ACM, 2013, pp. 17–24.
- [180] Marinai, S., Marino, E. & Soda, G., “Conversion of pdf books in epub format”, in: *2011 International Conference on Document Analysis and Recognition*, IEEE, 2011, pp. 478–482.
- [181] Martin, B., Koedinger, K. R., Mitrovic, A. & Mathan, S., “On using learning curves to evaluate its.”, in: *AIED*, 2005, pp. 419–426.
- [182] Martin, B., Mitrovic, A., Koedinger, K. R. & Mathan, S., “Evaluating and improving adaptive educational systems with learning curves”, *User Modeling and User-Adapted Interaction*, vol. 21, no. 3, 2011, pp. 249–283.
- [183] Martín-Moncunill, D., Sicilia-Urban, M.-Á., García-Barriocanal, E. & Sánchez-Alonso, S., “Evaluating the degree of domain specificity of terms in large terminologies”, *Online Information Review*, 2015.
- [184] Martinez-Rodriguez, J. L., Hogan, A. & Lopez-Arevalo, I., “Information extraction meets the semantic web: a survey”, *Semantic Web*, vol. 11, no. 2, 2020, pp. 255–335.
- [185] Matayoshi, J. & Lechuga, C., “Automated matching of its problems with textbook content.”, in: *Proceedings of the Second Workshop on Intelligent Textbooks*, CEUR-WS, 2020, pp. 17–28.
- [186] Mc Gurk, S., Abela, C. & Debattista, J., “Towards ontology quality assessment”, in: *Joint Proceedings of MEPDaW and LDQ 2017*, vol. 1824, CEUR-WS, 2017, pp. 94–106.
- [187] McCallum, A., “Information extraction: distilling structured data from unstructured text”, *Queue*, vol. 3, no. 9, 2005, pp. 48–57.

- [188] McDaniel, M. & Storey, V. C., “Evaluating domain ontologies: clarification, classification, and challenges”, *ACM Computing Surveys (CSUR)*, vol. 52, no. 4, 2019, pp. 1–44.
- [189] Medelyan, O., Witten, I. H. & Milne, D., “Topic indexing with wikipedia”, in: *Proceedings of the AAAI WikiAI workshop*, vol. 1, 2008, pp. 19–24.
- [190] Medford, A. J., Kunz, M. R., Ewing, S. M., Borders, T. & Fushimi, R., “Extracting knowledge from data through catalysis informatics”, *ACS Catalysis*, vol. 8, no. 8, 2018, pp. 7403–7429.
- [191] Melis, E., Gogvadze, G., Homik, M., Libbrecht, P., Ullrich, C. & Winterstein, S., “Semantic-aware components and services of activemath”, *British Journal of Educational Technology*, vol. 37, no. 3, 2006, pp. 405–423.
- [192] Mendes, P. N., Jakob, M., García-Silva, A. & Bizer, C., “Dbpedia spotlight: shedding light on the web of documents”, in: *Proceedings of the 7th international conference on semantic systems*, ACM, 2011, pp. 1–8.
- [193] Meng, R., Han, S., Huang, Y., He, D. & Brusilovsky, P., “Knowledge-based content linking for online textbooks”, in: *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, IEEE, 2016, pp. 18–25.
- [194] Merriam-Webster, *Knowledge*, <https://www.merriam-webster.com/dictionary/knowledge>, [Online; accessed 03-2022].
- [195] Mihalcea, R. & Csomai, A., “Wikify! linking documents to encyclopedic knowledge”, in: *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, 2007, pp. 233–242.
- [196] Milne, D., Medelyan, O. & Witten, I. H., “Mining domain-specific thesauri from wikipedia: a case study”, in: *2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06)*, IEEE, 2006, pp. 442–448.
- [197] Milne, D. & Witten, I. H., “Learning to link with wikipedia”, in: *Proceedings of the 17th ACM conference on Information and knowledge management*, ACM, 2008, pp. 509–518.
- [198] Mirizzi, R., Ragone, A., Di Noia, T. & Di Sciascio, E., “Ranking the linked data: The case of DBpedia”, in: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6189 LNCS, Springer, Berlin, Heidelberg, 2010, pp. 337–354, ISBN: 3642139108.
- [199] Mirizzi, R., Ragone, A., Di Noia, T. & Di Sciascio, E., “Semantic wonder cloud: exploratory search in dbpedia”, in: *International Conference on Web Engineering*, Springer, 2010, pp. 138–149.
- [200] Mirylenka, D., Passerini, A. & Serafini, L., “Bootstrapping domain ontologies from wikipedia: a uniform approach”, in: *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.
- [201] Mitrovic, A. & Ohlsson, S., “Evaluation of a constraint-based tutor for a database language”, *International Journal on Artificial Intelligence in Education*, vol. 10, 1999.
- [202] Mnih, A. & Kavukcuoglu, K., “Learning word embeddings efficiently with noise-contrastive estimation”, *Advances in neural information processing systems*, vol. 26, 2013.

- [203] Moore, J. W., “Are textbooks dispensable?”, *Journal of Chemical Education*, vol. 80, no. 4, 2003, p. 359.
- [204] Moro, A., Raganato, A. & Navigli, R., “Entity Linking meets Word Sense Disambiguation: a Unified Approach”, *Transactions of the Association for Computational Linguistics (TACL)*, vol. 2, 2014, pp. 231–244.
- [205] Mumpower, J. L. & Stewart, T. R., “Expert judgement and expert disagreement”, *Thinking & Reasoning*, vol. 2, no. 2-3, 1996, pp. 191–212.
- [206] Murray, M. C. & Pérez, J., “E-textbooks are coming: are we ready”, *Issues in Informing Science and Information Technology*, vol. 8, no. 6, 2011, pp. 49–60.
- [207] Naps, T. L., Eagan, J. R. & Norton, L. L., “Jhave – an environment to actively engage students in web-based algorithm visualizations”, in: *Thirty-first SIGCSE Technical Symposium on Computer Science Education*, vol. 32, ACM Press, 2000, pp. 109–113.
- [208] Navigli, R. & Ponzetto, S. P., “BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network”, *Artificial Intelligence*, vol. 193, 2012, pp. 217–250.
- [209] Newell, A. & Rosenbloom, P., “Mechanisms of skill acquisition”, *Cognitive skills and their acquisition*, 1981.
- [210] Ni, Y., Xu, Q. K., Cao, F., Mass, Y., Sheinwald, D., Zhu, H. J. & Cao, S. S., “Semantic documents relatedness using concept graph representation”, in: *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, 2016, pp. 635–644.
- [211] Nichols, T. R., Wisner, P. M., Cripe, G. & Gulabchand, L., “Putting the kappa statistic to use”, *The Quality Assurance Journal*, vol. 13, no. 3-4, 2010, pp. 57–61.
- [212] Oates, T., “Why textbooks count”, *A policy paper*, vol. 10, 2014.
- [213] Ogonowski, A., Herviou, M. L. & Dauphin, E., “Tools for extracting and structuring knowledge from texts”, in: *COLING 1994 Volume 2: The 15th International Conference on Computational Linguistics*, 1994.
- [214] Olson, D. L. & Delen, D., *Advanced data mining techniques*, Springer Science & Business Media, 2008.
- [215] Oro, E. & Ruffolo, M., “Pdf-trex: an approach for recognizing and extracting tables from pdf documents”, in: *2009 10th International Conference on Document Analysis and Recognition*, IEEE, 2009, pp. 906–910.
- [216] Parsons, D. & Haden, P., “Parson’s programming puzzles: a fun and effective learning tool for first programming courses”, in: *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52*, 2006, pp. 157–163.
- [217] Passant, A., “Measuring semantic distance on linking data and using it for resources recommendations.”, in: *AAAI spring symposium: linked data meets artificial intelligence*, vol. 77, 2010, p. 123.
- [218] Pelham, B. W., *Intermediate statistics: A conceptual course*, SAGE, 2013.
- [219] Pham, D. T. & Dimov, S. S., “An efficient algorithm for automatic knowledge acquisition”, *Pattern Recognition*, vol. 30, no. 7, 1997, pp. 1137–1143.

- [220] Piao, G., Ara, S. showkat & Breslin, J. G., “Computing the semantic similarity of resources in dbpedia for recommendation purposes”, in: *Joint International Semantic Technology Conference*, Springer, 2015, pp. 185–200.
- [221] Polpinij, J., “Ontology-based knowledge discovery from unstructured and semi-structured text”, PhD thesis, School of Computer Science and Software Engineering, University of Wollongong, 2014.
- [222] Porter, M. F., *Snowball: A language for stemming algorithms*, Published online, 2001, URL: <http://snowball.tartarus.org/texts/introduction.html>.
- [223] Rada, R., “Converting a textbook to hypertext”, *ACM Transactions on Information Systems*, vol. 10, no. 3, 1992, pp. 294–315.
- [224] 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, IEEE, 2003, pp. 87–91.
- [225] Rajman, M. & Besançon, R., “Text mining-knowledge extraction from unstructured textual data”, in: *Advances in data science and classification*, Springer, 1998, pp. 473–480.
- [226] Ramanathan, C., Jayabal, Y. & Sheth, M. J., “Challenges in generating bookmarks from toc entries in e-books”, in: *Proceedings of the 2012 ACM symposium on Document engineering*, vol. 37, 2012.
- [227] Ramnarayan, P., Tomlinson, A., Rao, A., Coren, M., Winrow, A. & Britto, J., “Isabel: a web-based differential diagnostic aid for paediatrics: results from an initial performance evaluation”, *Archives of disease in childhood*, vol. 88, no. 5, 2003, pp. 408–413.
- [228] Rau, L. F., Jacobs, P. S. & Zernik, U., “Information extraction and text summarization using linguistic knowledge acquisition”, *Information Processing & Management*, vol. 25, no. 4, 1989, pp. 419–428.
- [229] Rigutini, L., Di Iorio, E., Ernandes, M. & Maggini, M., “Automatic term categorization by extracting knowledge from the Web”, *Frontiers in Artificial Intelligence and Applications*, vol. 141, 2006, pp. 531–535.
- [230] Ritter, S., Fisher, J., Lewis, A., Bartle Finocchi, S., Hausmann, B. & Fancsali, S., “What’s a textbook? envisioning the 21st century k-12”, in: *Proceedings of the First Workshop on Intelligent Textbooks*, CEUR-WS, 2019, pp. 87–94.
- [231] Rivers, K., Harpstead, E. & Koedinger, K. R., “Learning curve analysis for programming: which concepts do students struggle with?”, in: *ICER*, vol. 16, 2016, pp. 143–151.
- [232] Rodríguez Rocha, O., Faron Zucker, C. & Giboin, A., “Extraction of relevant resources and questions from dbpedia to automatically generate quizzes on specific domains”, in: *Intelligent Tutoring Systems: 14th International Conference, ITS 2018, Montreal, QC, Canada, June 11–15, 2018, Proceedings 14*, Springer, 2018, pp. 380–385.
- [233] Rodríguez Rodríguez, J., Bruillard, E., Horsley, M., et al., *Digital textbooks, what’s new?*, IARTEM / Universidade de Santiago de Compostela, 2015, DOI: 10.15304/op377.759.
- [234] Rowley, J., “The wisdom hierarchy: representations of the dikw hierarchy”, *Journal of information science*, vol. 33, no. 2, 2007, pp. 163–180.

- [235] Ruiz Fabo, P., Bermúdez Sabel, H., Martínez Cantón, C. & González-Blanco, E., “The Diachronic Spanish Sonnet Corpus (DISCO): TEI and Linked Open Data Encoding, Data Distribution and Metrical Findings”, in: *Digital Humanities 2018, Book of Abstracts*, 2018.
- [236] Rusu, O., Halcu, I., Grigoriu, O., Neculoiu, G., Sandulescu, V., Marinescu, M. & Marinescu, V., “Converting unstructured and semi-structured data into knowledge”, in: *2013 11th RoEduNet International Conference*, IEEE, 2013, pp. 1–4.
- [237] Sakimura, N., Bradley, J., Jones, M., De Medeiros, B. & Mortimore, C., “Openid connect core 1.0”, *The OpenID Foundation*, 2014, S3.
- [238] Salton, G., Wong, A. & Yang, C. S., “A vector space model for automatic indexing”, *Commun. ACM*, vol. 18, no. 11, Nov. 1975, pp. 613–620.
- [239] Salton, G. & Buckley, C., “Term-weighting approaches in automatic text retrieval”, *Information processing & management*, vol. 24, no. 5, 1988, pp. 513–523.
- [240] Salzberg, S. L., “On comparing classifiers: pitfalls to avoid and a recommended approach”, *Data mining and knowledge discovery*, vol. 1, no. 3, 1997, pp. 317–328.
- [241] Savenkov, V., Mehmood, Q., Umbrich, J. & Polleres, A., “Counting to k or how sparql 1 property paths can be extended to top-k path queries”, in: *Proceedings of the 13th international conference on semantic systems*, 2017, pp. 97–103.
- [242] Schreiber, G., “Knowledge engineering”, *Foundations of Artificial Intelligence*, vol. 3, 2008, pp. 929–946.
- [243] Seaborne, A. & Harris, S., *SPARQL 1.1 Query Language*, W3C Recommendation, W3C, Mar. 2013.
- [244] Severance, C. R., *Python for everybody exploring data using Python 3*, 2016.
- [245] Shaffer, C., “Opensda: an interactive etextbook for computer science courses”, in: *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, ACM, 2016, pp. 5–5, ISBN: 978-1-4503-3685-7.
- [246] Shanmugam, R. & Chattamvelli, R., *Statistics for scientists and engineers*, Wiley, 2015.
- [247] Shao, M. & Futrelle, R. P., “Recognition and classification of figures in pdf documents”, in: *International Workshop on Graphics Recognition*, Springer, 2005, pp. 231–242.
- [248] Sint, R., Schaffert, S., Stroka, S. & Ferstl, R., “Combining unstructured, fully structured and semi-structured information in semantic wikis”, in: *CEUR Workshop Proceedings*, vol. 464, Heraklion Crete, Greece, 2009, pp. 73–87.
- [249] Slabbekoorn, K., Hollink, L. & Houben, G.-J., “Domain-aware ontology matching”, in: *The Semantic Web–ISWC 2012: 11th International Semantic Web Conference*, Springer, 2012, pp. 542–558.
- [250] Soderland, S., “Learning information extraction rules for semi-structured and free text”, *Machine learning*, vol. 34, no. 1, 1999, pp. 233–272.
- [251] Sosnovsky, S., Dietrich, M., Andrès, E., Gogvadze, G., Winterstein, S., Libbrecht, P., Siekmann, J. & Melis, E., “Math-bridge: closing gaps in european remedial mathematics with technology-enhanced learning”, in: *Mit werkzeu-*

- gen mathematik Und stochastik lernen—using tools for learning mathematics and statistics*, Springer, 2014, pp. 437–451.
- [252] Sosnovsky, S. & Brusilovsky, P., “Evaluation of topic-based adaptation and student modeling in quizguide”, *User Modeling and User-Adapted Interaction*, vol. 25, no. 4, 2015, pp. 371–424.
 - [253] Sosnovsky, S., Dolog, P., Henze, N., Brusilovsky, P. & Nejdl, W., “Translation of overlay models of student knowledge for relative domains based on domain ontology mapping”, in: *Proceedings of the 2007 Conference on Artificial Intelligence in Education*, NLD: IOS Press, 2007, 289–296.
 - [254] Sosnovsky, S., Hsiao, I.-H. & Brusilovsky, P., “Adaptation “in the wild”: ontology-based personalization of open-corpus learning material”, in: *European Conference on Technology Enhanced Learning*, Springer, 2012, pp. 425–431.
 - [255] Stahn, L.-L., Hennicke, S. & De Luca, E. W., “Using tei for textbook research”, in: *Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities*, 2016, pp. 181–186.
 - [256] Stark, P. B., *Glossary of Statistical Terms from SticiGui*, <https://www.stat.berkeley.edu/~stark/SticiGui/Text/gloss.htm>, [Online; accessed 12-2020], 2019.
 - [257] Tacoma, S., Sosnovsky, S., Boon, P., Jeuring, J. & Drijvers, P., “The interplay between inspectable student models and didactics of statistics”, *Digital Experiences in Mathematics Education*, vol. 4, no. 2, 2018, pp. 139–162.
 - [258] Tang, Y. Y., De Yan, C. & Suen, C. Y., “Document processing for automatic knowledge acquisition”, *IEEE transactions on Knowledge and Data Engineering*, vol. 6, no. 1, 1994, pp. 3–21.
 - [259] Tankeleviciene, L. & Damasevicius, R., “Characteristics of domain ontologies for web based learning and their application for quality evaluation”, *Informatics in Education*, vol. 8, no. 1, 2009, pp. 131–152.
 - [260] Thaker, K., Brusilovsky, P. & He, D., “Student modeling with automatic knowledge component extraction for adaptive textbooks.”, in: *Proceedings of the First Workshop on Intelligent Textbooks*, vol. 2384, CEUR-WS, 2019, pp. 95–102.
 - [261] 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)*, IEEE, 2018, pp. 413–420.
 - [262] Thareja, R., V., V. & Mohania, M., “Pdf2prereq: automatic extraction of concept dependency graphs from academic textbooks”, in: *AAAI2022 Artificial Intelligence for Education*, 2022.
 - [263] *The Chicago Manual of Style*, 17th ed., The University of Chicago Press, 2017.
 - [264] Tittel, S., Bermúdez-Sabel, H. & Chiarcos, C., “Using RDFa to Link Text and Dictionary Data for Medieval French”, in: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*, 2018, ISBN: 979-10-95546-19-1.
 - [265] Tkaczyk, D., Szostek, P., Fedoryszak, M., Dendek, P. J. & Bolikowski, Ł., “Cermine: automatic extraction of structured metadata from scientific liter-

- ature”, *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 18, no. 4, 2015, pp. 317–335.
- [266] Tran, T. T., Miwa, M. & Ananiadou, S., “Syntactically-informed word representations from graph neural network”, *Neurocomputing*, vol. 413, 2020, pp. 431–443.
- [267] Trella, M., Carmona, C. & Conejo, R., “Medea: an open service-based learning platform for developing intelligent educational systems for the web”, in: *Workshop on Adaptive Systems for Web-Based Education: tools and reusability (AIED’05)*, 2005, pp. 27–34.
- [268] Tuarob, S., Bhatia, S., Mitra, P. & Giles, C. L., “Automatic detection of pseudocodes in scholarly documents using machine learning”, in: *2013 12th International Conference on Document Analysis and Recognition*, IEEE, 2013, pp. 738–742.
- [269] Tuarob, S., Bhatia, S., Mitra, P. & Giles, C. L., “Algorithmseer: a system for extracting and searching for algorithms in scholarly big data”, *IEEE Transactions on Big Data*, vol. 2, no. 1, 2016, pp. 3–17.
- [270] Tuarob, S., Mitra, P. & Giles, C. L., “A hybrid approach to discover semantic hierarchical sections in scholarly documents”, in: *2015 13th international conference on document analysis and recognition (ICDAR)*, IEEE, 2015, pp. 1081–1085.
- [271] Turban, E., Rainer, R. K., Potter, R. E., et al., *Introduction to information technology*, John Wiley & Sons New York, NY, 2001.
- [272] Ubøe, J., *Introductory statistics for business and economics: theory, exercises and solutions*, Springer International Publishing AG, 2017.
- [273] Ullrich, C. & Melis, E., “Pedagogically founded courseware generation based on htn-planning”, *Expert Systems with Applications*, vol. 36, no. 5, 2009, pp. 9319–9332.
- [274] University of St Andrews, *Statistics Glossary*, <https://web.archive.org/web/20161113233144/http://www.st-andrews.ac.uk/psychology/current/statisticsglossary/#d.en.78939>, [Online; accessed 12-2020], 2016.
- [275] Usbeck, R., Ngomo, A. C. N., Röder, M., Gerber, D., Coelho, S. A., Auer, S. & Both, A., “AGDISTIS - Graph-based disambiguation of named entities using linked data”, in: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8796, 2014, pp. 457–471, ISBN: 9783319119632.
- [276] Vanderplas, J. T., *Python data science handbook: essential tools for working with data*, 1st, O’Reilly, 2017.
- [277] Velenik, Y., *Probabilités et statistique*, Université de Genève, 2012.
- [278] Vivaldi, J. & Rodríguez, H., “Finding domain terms using wikipedia”, in: *Proceedings of the 7th International Conference on Language Resources and Evaluation, LREC 2010*, 2010, pp. 386–393, ISBN: 2951740867.
- [279] Vrandečić, D. & Krötzsch, M., “Wikidata: a free collaborative knowledge-base”, *Commun. ACM*, vol. 57, no. 10, Sept. 2014, pp. 78–85.

- [280] Wali, A., Chun, S. A. & Geller, J., “A bootstrapping approach for developing a cyber-security ontology using textbook index terms”, in: *2013 International Conference on Availability, Reliability and Security*, IEEE, 2013, pp. 569–576.
- [281] Walpole, R. E., *Probabilidad estadística para ingenieros*, Pearson, 2012.
- [282] Walpole, R. E., Myers, R. H., Myers, S. L., et al., *Probability & statistics for engineers & scientists*, Prentice Hall, 2012.
- [283] Wang, M., Chau, H., Thaker, K., Brusilovsky, P. & He, D., “Concept annotation for intelligent textbooks”, *arXiv preprint arXiv:2005.11422*, 2020.
- [284] Wang, S., Liang, C., Wu, Z., Williams, K., Pursel, B., Brautigam, B., Saul, S., Williams, H., Bowen, K. & Giles, C. L., “Concept hierarchy extraction from textbooks”, in: *Proceedings of the 2015 ACM Symposium on Document Engineering*, ACM, 2015, pp. 147–156.
- [285] Wang, S., Ororbia, A., Wu, Z., Williams, K., Liang, C., Pursel, B. & Giles, C. L., “Using prerequisites to extract concept maps from textbooks”, in: *Proceedings of the 25th acm international on conference on information and knowledge management*, 2016, pp. 317–326.
- [286] Weber, G., “Cognitive diagnosis and episodic modelling in an intelligent lisp-tutor”, in: *Intelligent Tutoring Systems*, ed. by C., F., artc106: Univ. of Montreal, 1988, pp. 207–214.
- [287] Whittington, J., “Pdf explained”, in: O'Reilly and Associate Series, O'Reilly Media, 2011, chap. I Introduction, ISBN: 9781449310028.
- [288] Winchell, A., Mozer, M., Lan, A., Grimaldi, P. & Pashler, H., “Can textbook annotations serve as an early predictor of student learning?”, *International Educational Data Mining Society*, 2018.
- [289] Witten, I. H. & Milne, D. N., “An effective, low-cost measure of semantic relatedness obtained from wikipedia links”, in: *Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy*, AAAI Press, 2008.
- [290] Wu, J., Williams, K. M., Chen, H.-H., Khabsa, M., Caragea, C., Tuarob, S., Ororbia, A. G., Jordan, D., Mitra, P. & Giles, C. L., “Citeseerx: ai in a digital library search engine”, *AI Magazine*, vol. 36, no. 3, 2015, pp. 35–48.
- [291] Wu, Z., Das, S., Li, Z., Mitra, P. & Giles, C. L., “Searching online book documents and analyzing book citations”, in: *Proceedings of the 2013 ACM symposium on Document engineering*, 2013, p. 81, ISBN: 9781450317894.
- [292] Wu, Z., Li, Z., Mitra, P. & Giles, C. L., “Can back-of-the-book indexes be automatically created?”, in: *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, ACM, 2013, pp. 1745–1750.
- [293] Wu, Z., Mitra, P. & Giles, C. L., “Table of contents recognition and extraction for heterogeneous book documents”, in: *2013 12th international conference on document analysis and recognition*, IEEE, 2013, pp. 1205–1209.
- [294] Xu, F., Kurz, D., Piskorski, J. & Schmeier, S., “A domain adaptive approach to automatic acquisition of domain relevant terms and their relations with bootstrapping.”, in: *LREC*, 2002.
- [295] Yao, Y., Zeng, Y., Zhong, N. & Huang, X., “Knowledge retrieval (kr)”, in: *IEEE/WIC/ACM International Conference on Web Intelligence (WI'07)*, IEEE, 2007, pp. 729–735.

- [296] Yuan, X., Wang, T., Meng, R., Thaker, K., Brusilovsky, P., He, D. & Trischler, A., “One size does not fit all: generating and evaluating variable number of keyphrases”, in: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, July 2020.
- [297] Yuan, Y., Gao, J. & Zhang, Y., “Supervised learning for robust term extraction”, in: *2017 International Conference on Asian Language Processing (IALP)*, IEEE, 2017, pp. 302–305.
- [298] Yudelson, M. & Brusilovsky, P., “Navex: providing navigation support for adaptive browsing of annotated code examples”, in: *12th International Conference on Artificial Intelligence in Education, AI-Ed’2005*, ed. by Looi, C.-K., McCalla, G., Bredeweg, B. & Breuker, J., IOS Press, 2005, pp. 710–717.
- [299] Zhang, J., Liu, M. & Zhang, Y., “Topic-informed neural approach for biomedical event extraction”, *Artificial intelligence in medicine*, vol. 103, 2020, p. 101783.
- [300] Zhang, Y., Chen, F., Zhang, W., Zuo, H. & Yu, F., “Keywords extraction based on word2vec and textrank”, in: *Proceedings of the 2020 The 3rd International Conference on Big Data and Education*, 2020, pp. 37–42.
- [301] Zhou, X., Han, H., Chankai, I., Prestrud, A. A. & Brooks, A. D., “Converting semi-structured clinical medical records into information and knowledge”, in: *21st International Conference on Data Engineering Workshops (ICDEW’05)*, IEEE, 2005, pp. 1162–1162.
- [302] Zhu, G. & Iglesias, C. A., “Exploiting semantic similarity for named entity disambiguation in knowledge graphs”, *Expert Systems with Applications*, vol. 101, 2018, pp. 8–24.

Summary

Many adaptive educational systems and other artificial intelligence applications rely on high-quality knowledge representations. Still, knowledge acquisition remains the primary bottleneck hindering large-scale deployment and adoption of knowledge-based systems. The manual creation of knowledge models requires a lot of time and effort. One path to scalable knowledge extraction is using digital textbooks, given their domain-oriented content, structure, and availability. The authors' knowledge encoded in the textbooks' elements that facilitate navigation and understanding of the material (table of contents, index, formatting styles) can be leveraged to generate knowledge models. Nevertheless, extracting this hidden knowledge from digital (PDF) textbooks is challenging. This dissertation presents a unified approach for automatically extracting high-quality and domain-specific knowledge models from digital textbooks.

This dissertation's approach extracts an initial set of information (structure, content, and terminology) from the textbooks, then gradually adds new information (links and semantic content), and finally analyses and refines the knowledge about the domain (concepts). The proposed approach consists of seven phases, each focusing on different aspects of knowledge that can be extracted from textbooks. Additionally, multiple evaluations verify the *quality* of the extracted models.

The first phase of the approach, extraction (📌 Chapter 3), describes multiple steps to automatically recognize the structure, content, and domain terms in textbooks. Structural information refers to the list of chapters and subchapters of the textbook. The textbook's content is represented hierarchically (words, lines, text fragments, pages, and sections). Lastly, the domain terms are extracted from the back-of-the-book index. The extracted elements are recognized with a very high *accuracy*: almost absolute precision and recall for the structure, higher recognition for the content than state-of-the-art tools (e.g., 97% vs. 85%), and very high precision and recall for the domain terms (between 96% and 99%).

Linking/enrichment is the second phase of the approach (📌 Chapter 4), where the domain terms are used as a bridge to connect the textbooks to an external knowledge graph. Specifically, domain terms are matched to corresponding entities in DBpedia—a publicly available knowledge graph based on Wikipedia. The linking mechanism achieves high and balanced precision and recall values (e.g., 97% and 92%, respectively), which allows for the enrichment of the domain terms with the *semantic information* (abstracts, Wikipedia links, categories, synonyms, and relations

to other terms) that matches the terms' actual meanings ("sense").

Domain analysis is performed in the third, fourth, and fifth phases. The third phase of the approach, integration (🔗 Chapter 4), describes integrating the domain terms from multiple textbooks into a single model by merging semantically equal terms. When more textbooks from the target domain are combined, the *coverage* of the domain grows significantly. For example, one-third of the target domain while using ten textbooks vs. less than one-tenth using only one textbook.

Categorization (🔗 Chapter 5) is the fourth phase of the approach. It details the steps to identify the relevance of concepts to the target domain. The information from DBpedia (categories, links, and abstracts) and textbooks (index terms and domain information) is used to identify how individual concepts are related to the target domain: most essential concepts in the target domain, other concepts in the target domain, concepts in neighboring domains, and concepts unrelated to the target domain. Distinguishing the relevance of the concepts to the target domain, i.e., their *specificity*, is achieved with high accuracy (e.g., 92%).

The fifth phase, validation (🔗 Chapter 6), describes an experiment that analyses the cognitive validity of textbook concepts for knowledge modeling using learning curve analysis. The results show that, generally, textbook concepts are *cognitively valid* knowledge units (learning takes place during student practicing for 44 out of 46 studied concepts). Additionally, the experiment demonstrates that in terms of *granularity*, fine-grained concepts model knowledge better.

The sixth phase of the automatic approach is formalization (🔗 Chapter 3), where all the extracted knowledge is serialized as a descriptive XML file using the Text Encoding Initiative.

Finally, after the approach has produced knowledge models, they can be used in various applications. This dissertation introduces three educational systems that use the extracted models at their core (🔗 Chapter 7). (1) Interlingua uses the knowledge models as a semantic bridge between textbooks written in different languages. (2) Intextbooks supports the adaptation and interactivity of all possible elements in HTML textbooks thanks to the fine-grained identification of content elements in the extracted knowledge models. (3) The integration of Intextbooks with P^4 —a personalized Python programming practice system—shows how the knowledge models can facilitate linking between textbook content and smart learning activities.

Multiple domains are used for the evaluation of the proposed approach. Extraction of the textbook content is tested in the domains of *statistics*, *computer science*, *history*, and *literature*. Terms in the domains of *statistics* and *information retrieval* are linked correctly to their respective entities in DBpedia. The specificity of terms in *statistics* and *ancient philosophy* is identified. Finally, concepts extracted from *Python programming* textbooks are shown to be valid knowledge components.

In conclusion, this dissertation explores and presents an approach for automatically extracting knowledge models from digital textbooks taking into account multiple quality aspects. Future research involves extending and further evaluating the approach, evaluating the extracted knowledge models from the prospect of pedagogical usefulness, and extending the use of the approach toward creating intelligent textbooks.

Samenvatting

Veel adaptieve onderwijssystemen en andere toepassingen van kunstmatige intelligentie vertrouwen op hoogwaardige kennisrepresentaties. Kennisverwerving is het belangrijkste knelpunt dat grootschalige implementatie en acceptatie van op kennis gebaseerde systemen in de weg staat. Het handmatig aanmaken van kennismodellen kost veel tijd en inspanning. Een manier om schaalbare kennisextractie te bereiken is het gebruik van digitale studieboeken, gezien hun domeinspecifieke inhoud, structuur en beschikbaarheid. De kennis van de auteurs die is gecodeerd in de elementen van de studieboeken die navigatie en begrip van het materiaal vergemakkelijken (inhoudsopgave, index, opmaakstijlen) kan worden gebruikt om kennismodellen te genereren. Desalniettemin is het een uitdaging om deze verborgen kennis uit digitale (pdf) studieboeken te halen. Dit proefschrift presenteert een uniforme aanpak voor het automatisch extraheren van hoogwaardige en domeinspecifieke kennismodellen uit digitale studieboeken.

De methode die in het proefschrift wordt beschreven, haalt een eerste set informatie (structuur, inhoud en terminologie) uit de studieboeken, voegt vervolgens geleidelijk nieuwe informatie toe (links en semantische inhoud), en analyseert en verfijnt ten slotte de kennis over het domein (concepten). De voorgestelde aanpak bestaat uit zeven fasen, die zich elk richten op verschillende aspecten van kennis die uit studieboeken kunnen worden gehaald. Daarnaast verifiëren meerdere evaluaties de *kwaliteit* van de geëxtraheerde modellen.

De eerste fase van de aanpak, extractie (📖 Hoofdstuk 3), beschrijft meerdere stappen om de structuur, inhoud en domeintermen in studieboeken automatisch te herkennen. Met structurele informatie wordt de lijst met hoofdstukken en subhoofdstukken van het studieboek bedoeld. De inhoud van het studieboek wordt hiërarchisch weergegeven (woorden, regels, tekstfragmenten, pagina's en paragrafen). De domeintermen uit de index aan het einde van het boek gehaald. De geëxtraheerde elementen worden herkend met een zeer hoge *nauwkeurigheid*: bijna volledige precisie en recall voor de structuur, betere herkenning van de inhoud dan state-of-the-art tools (bijv. 97% vs. 85%), en zeer hoge precisie en recall voor de domeintermen (tussen 96% en 99%).

Koppelen/verrijken is de tweede fase van de aanpak (📖 Hoofdstuk 4), waarbij de domeintermen worden gebruikt als een brug om de studieboeken te verbinden met een externe kennisgraaf. In het bijzonder worden domeintermen gekoppeld aan overeenkomstige entiteiten in DBpedia, een openbaar beschikbare kennis-

graaf op basis van Wikipedia. Het koppelingsmechanisme bereikt hoge en gebalanceerde precisie en recall-waarden (bijv. 97% respectievelijk 92%), wat het mogelijk maakt de domeintermen te verrijken met de *semantische informatie* (samenvattingen, Wikipedia-links, categorieën, synoniemen en relaties met andere termen) die overeenkomen met de werkelijke betekenis van de termen ("sense").

Domeinanalyse wordt uitgevoerd in de derde, vierde en vijfde fase. De derde fase van de aanpak, integratie (🔗 Hoofdstuk 4), beschrijft het integreren van de domeintermen uit meerdere studieboeken in een enkel model door semantisch gelijke termen samen te voegen. Wanneer meer studieboeken uit het doeldomein worden gecombineerd, groeit de *dekking* van het domein aanzienlijk. Bijvoorbeeld een derde van het doeldomein met tien studieboeken vs. minder dan een tiende met slechts één studieboek.

Categorisatie (🔗 Hoofdstuk 5) is de vierde fase van de aanpak. Het beschrijft de stappen om de relevantie van concepten voor het doeldomein te identificeren. De informatie uit DBpedia (categorieën, links en samenvattingen) en studieboeken (indextermen en domeininformatie) wordt gebruikt om vast te stellen hoe individuele concepten verband houden met het doeldomein: de meest essentiële concepten in het doeldomein, andere concepten in het doeldomein, concepten in aangrenzende domeinen en concepten die geen verband houden met het doeldomein. Het onderscheiden van de relevantie van de concepten voor het doeldomein, d.w.z. hun *specificiteit*, wordt bereikt met een hoge nauwkeurigheid (bijv. 92%).

De vijfde fase, validatie (🔗 Hoofdstuk 6), beschrijft een experiment dat de cognitieve validiteit van studieboekconcepten voor kennismodellering analyseert met behulp van leercurve-analyse. De resultaten laten zien dat studieboekconcepten over het algemeen *cognitief geldige* kenniseenheden zijn (tijdens het oefenen van studenten vindt leren plaats voor 44 van de 46 bestudeerde concepten). Bovendien toont het experiment aan dat in termen van *granulariteit* fijnmazige concepten kennis beter modelleren.

De zesde fase van de automatische aanpak is formalisatie (🔗 Hoofdstuk 3), waarbij alle geëxtraheerde kennis wordt geserialiseerd als een XML-bestand met behulp van het Text Encoding Initiative.

Als de aanpak uiteindelijk kennismodellen heeft opgeleverd, kunnen deze in verschillende toepassingen worden gebruikt. Dit proefschrift introduceert drie educatieve systemen die de geëxtraheerde modellen gebruiken (🔗 Hoofdstuk 7). (1) Interlingua gebruikt de kennismodellen als een semantische brug tussen studieboeken die in verschillende talen zijn geschreven. (2) Intextbooks ondersteunt de aanpassing en interactiviteit van alle mogelijke elementen in HTML-studieboeken dankzij de fijnmazige identificatie van inhoudselementen in de geëxtraheerde kennismodellen. (3) De integratie van Intextbooks met P^4 —een gepersonaliseerd oefensysteem voor programmeren in Python—laat zien hoe de kennismodellen de koppeling tussen de inhoud van studieboeken en slimme leeractiviteiten makkelijker kunnen maken.

Voor de evaluatie van de voorgestelde aanpak worden meerdere domeinen gebruikt. De extractie van de inhoud van studieboeken wordt getest in de domeinen *statistiek*, *informatica*, *geschiedenis* en *literatuur*. Termen in de domeinen *statistiek* en *information retrieval* zijn correct gekoppeld aan hun respectievelijke entiteiten in DBpedia. De specificiteit van termen in de *statistiek* en *antieke filosofie* wordt geï-

dentificeerd. Ten slotte wordt aangetoond dat concepten die zijn geëxtraheerd uit studieboeken over *Python programmeren* geldige kenniscomponenten zijn.

Concluderend, dit proefschrift onderzoekt en presenteert een aanpak voor het automatisch extraheren van kennismodellen uit digitale studieboeken, rekening houdend met meerdere kwaliteitsaspecten. Toekomstig onderzoek omvat het uitbreiden en verder evalueren van de aanpak, het evalueren van de geëxtraheerde kennismodellen vanuit het perspectief van pedagogische bruikbaarheid, en het uitbreiden van het gebruik van de aanpak naar het maken van intelligente studieboeken.

Acknowledgments

After finishing my master's in 2012, I decided to do a PhD. However, I needed a break before embarking on such a journey. At that moment, I thought I would return to Costa Rica and start looking for a PhD in a year or two. It took me six years to finally start this adventure.

First, I want to thank my supervisor, Sergey. We met in 2010 in Saarbrücken, Germany. We worked together at the DFKI, and he was the supervisor of my master's thesis. In 2016, we met again in Costa Rica because of a conference. At that moment, Sergey told me he would start a position at Utrecht University, and we discussed the possibility of me joining for a PhD. After looking for funding, I finally started in June 2018. I thank Sergey for his guidance, always having time for meetings and discussions, and always pushing me to aim high.

My PhD would not have been possible without a promotor. I want to thank Johan for hosting me. Thank you for your guidance, support, and always having time for questions and nice chats. Also, I want to thank the Assessment Committee (Judith Masthoff, Kees van Deemter, Vania Dimitrova, Jacopo Urbani, and Peter Brusilovsky) for taking the time to review my thesis. Special thanks to Peter for collaborating with me on multiple works; I have learned from all your experience and knowledge. Also, thanks to PAWS members and collaborators: Jordan, Kamil, Khushboo, and Alireza.

Thanks to *Ministerio de Ciencia, Innovación, Tecnología y Telecomunicaciones* (MICITT), Costa Rica for financing this research. Also, to all my colleagues at *Administración de Tecnología de Información* (ATI) from the *Instituto Tecnológico de Costa Rica* (TEC) for supporting my academic development.

Within the UU, I was part of the STLT group. Thank you Johan, Sergey, Matthieu, Hieke, Ioanna, Arjan, Raja, Nico, Qixiang, Aditja, Almed, Niek, and Eduardo for making our group so friendly and welcoming. I also want to thank my other colleagues that have been part of this journey: Vedran, Samira, Saba, Jacco, Imke, Wishnu, Anna-Lenna, Pablo, Marijn, Gabi, and many more.

I am very grateful that I didn't start this journey alone. Thanks, José for being there. Moving to another country with two cats and several suitcases was an adventure for both of us. I know it was not always easy for you, but I also know that we had nice times. Thank you for your support and affection. This PhD could not have been possible without you in the equation. I can finally use the engraved pen that you gave me just at the start.

Having a really good friend in a new country makes life better. In my case, that

friend is Vedran. After a few months of starting my PhD, I moved to his office. After that, we started talking more and more, and without knowing, we became friends. I started drinking coffee thanks to Vedran. Despite growing up in a coffee-producer country, I never drank it. Before covid, we used to have coffee breaks in the afternoon at the office to talk and relax before going back to our research. Those breaks made our working days less long. Thank you Vedran, and Elena, for the countless gatherings for home-cook dinners, watching movies, playing games, eating out, or simply talking. Samira and I were also office mates. Sadly, we did not share that much in the office because of covid. However, in the last few months, we have become good friends. Thank you for all the nice conversations, lunches/dinners, recommendations, and exchanging of stories from our home countries. I thank you both for your friendship and for being my paranymphs.

Camila is a very good friend that I made while living in Germany. We arrived at the same time in Frankfurt, and we discovered the new country together. When I moved back to Europe, I immediately contacted her to see each other again. Now, every year (except during covid, of course), we see each other in Germany during December to visit the Christmas markets. Thank you, Julio, for also giving me your friendship. And, of course, a special hug for the new member of their family, Emilia.

Despite the distance, my friends in Costa Rica stayed in touch all these years. I talked with Isaura almost daily, commenting about our days and plans for the weekend. I asked for her opinion and help multiple times while designing pictures for my papers and presentations. In fact, she designed the nice cover of this thesis. She was going to come for a visit in 2020, but covid forced us to cancel. Isaura, thank you for being such a good friend. Thank you Anita, Iván, and Isaura, for almost every day saying "good morning" in our "Random" group chat. Thanks to Esteban, Alberto, and Andrés for keeping our friendship intact and checking in occasionally. Thanks to Maikol for being there in the happiest and toughest moments. My oldest friendship is with him, and it is a great gift. Thanks to Laurita for always caring about me.

Live in the Netherlands has been so nice because of all the dinners, movies, games, drinks, and stories that I have shared with friends: Vedran, Elena, Samira, Eduardo, Saba, Adi, Neha, Aina, Robert, Qixiang, Christian, Zulema, Kiko, René, Taco, Saskia, Anja, Paul, Carla, Roos, Marieke, Lukas, Iris, Julian, Anke, and Menno.

During a PhD, having a support system is very important. In my case, it came from four-legged friends (a.k.a *gatitos*). Máxima and Museo came with me to the Netherlands, and by now, they have lived more here than in Costa Rica. Museo is by far my buddy. He kept me company, in a comfy place between the keyboard and the monitor, during the long nights working on papers. I think he also deserves a PhD. Máxima distracted me in the most stressful moments when I would hear her meow with a toy in her mouth—her signal for playtime and attention. However, she usually went to bed early, even if Museo and I were at the computer (she doesn't deserve a PhD). I am also lucky to have gained two more *gatitos*—Max and Billy. Max has so much energy and is so childish that it is impossible not to laugh and enjoy his personality. Billy is the most adorable and chill cat ever. Just stopping for a moment and observing Billy's inner peace was enough for me to slow down and enjoy the present. In the relaxing times, having Maxi and Billy boy on the sofa made those moments even better. The three boys and the lady have made my life and my

PhD, so much more special and enjoyable.

Finally, I want to thank Jeroen—thank you for joining my life. Thank you for all your support and for being a great team member. Finishing a PhD is a lot of work, but we always find nice activities and experiences to enjoy life despite the daily chaos. Laughter is the ultimate stress reliever, and in the *dorstige harthuis*, there is always a big dose of that! (from SSP to BH).

For my family. *Mami, muchas gracias por siempre apoyarme y desear mi felicidad. Estar tan lejos no ha sido fácil, pero con el cariño y amor que nos tenemos hemos podido seguir adelante. Gracias por todo el amor, educación, y esfuerzo que me distes siempre. Gracias a ello, he podido alcanzar este y muchos más logros. Sería imposible haber hecho este viaje sin contar con el apoyo de mi hermano José. Muchas gracias por llevar la administración de todos los asuntos de la casa, y por estar siempre ahí. Los quiero mucho a los dos.*

Curriculum Vitae

Isaac Alpizar Chacon

Born on 07 April 1987 in Alajuela, Costa Rica

Education

- 2018 – 2023** Doctor of Philosophy (PhD) in Computer Science
Universiteit Utrecht, Netherlands
- 2010 – 2012** Master of Science (MSc) in Computer Science
Universität des Saarlandes, Germany
- 2005 – 2009** Bachelor Degree in Computer Engineering (with honor)
Instituto Tecnológico de Costa Rica, Costa Rica

Work Experience

- 2014 – date** Adjunct Teacher
Instituto Tecnológico de Costa Rica, Costa Rica
- 2012 – 2018** Software Engineer
Instituto Tecnológico de Costa Rica, Costa Rica
- 2010 – 2012** Student Research Assistant (HiWi)
*German Research Center for Artificial Intelligence (DFKI),
Germany*
- 2008 – 2010** Software Engineer
Instituto Tecnológico de Costa Rica, Costa Rica

