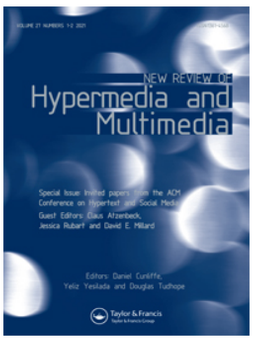

New Review of Hypermedia and Multimedia

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tham20

Knowledge models from PDF textbooks

Isaac Alpizar-Chacon & Sergey Sosnovsky

To cite this article: Isaac Alpizar-Chacon & Sergey Sosnovsky (2021) Knowledge models from PDF textbooks, New Review of Hypermedia and Multimedia, 27:1-2, 128-176, DOI: 10.1080/13614568.2021.1889692

To link to this article: https://doi.org/10.1080/13614568.2021.1889692

Published online: 28 Feb 2021.

NEW REVIEW OF HYPERMEDIA AND MULTIMEDIA
2021, VOL. 27, NOS. 1–2, 128–176
https://doi.org/10.1080/13614568.2021.1889692

Knowledge models from PDF textbooks

Isaac Alpizar-Chacon and Sergey Sosnovsky

Utrecht University, Utrecht, The Netherlands

ABSTRACT

Textbooks are educational documents created, structured and formatted by domain experts with the primary 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. Our paper presents an extensible approach towards automated extraction of knowledge models from textbooks and enrichment of their content with additional links (both internal and external). The textbooks themselves essentially become hypertext documents where individual pages are annotated with important concepts in the domain. The evaluation experiments examine several aspects and stages of the approach, including the accuracy of model extraction, the pragmatic quality of extracted models using one of their possible applications— semantic linking of textbooks in the same domain, the accuracy of linking models to external knowledge sources and the effect of integration of multiple textbooks from the same domain. The results indicate high accuracy of model extraction on symbolic, syntactic and structural levels across textbooks and domains, and demonstrate the added value of the extracted models on the semantic level.

ARTICLE HISTORY
Received 5 June 2020
Accepted 9 February 2021

KEYWORDS
Textbook; PDF processing; model extraction; knowledge modelling; named entity disambiguation; DBpedia; semantic linking

1. Introduction

Textbooks are high-quality textual resources. Content of a typical textbook is characterised 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.

CONTACT Isaac Alpizar-Chacon i.alpizarchacon@uu.nl

For the purpose of text analysis and information extraction, textbooks have often been considered as high-quality but non-structured information resources (Ramnarayan et al., 2003). 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 (Chambliss, 2002). It has been shown that text structure awareness is an important foundation for improving text comprehension and recall by learners (Hall et al., 2005).

Hence, the content of a good textbook is structured into sections and subsections 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 (Table of Content) 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 paper 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. Additionally, we take the next stage and demonstrate how these models can automatically link textbooks from the same domain and integrate textbooks with a global reference model, such as DBpedia.[1] As a result, textbooks can potentially become ubiquitous sources of knowledge for a growing Linked Open Data initiative.[2] Textbooks themselves also transform into collections of documents linked to this Linked Open Data cloud and become available for a rich variety of services that can facilitate their discovery, enrichment, adaptation, etc.

We focus 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. we have processed EPUB textbooks as well). A typical PDF document contains thousands of low-level objects, multiple compression mechanisms, different font formats, lines, curves, vectors, and ancillary content (Whitington, 2011). 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

[1] http://dbpedia.org
[2] https://lod-cloud.net/

preserve information that would facilitate extraction of their structure (i.e. letters composing words, words belonging to paragraphs, organisation of lists, tables and figures, hierarchy of sections, or even the reading order of the text) (Tkaczyk et al., 2015). The PDF standard does allow the expression of a document's logical structure with tags, but their usage is seldom and inconsistent; therefore, our approach does not rely on tagging. Instead, we have developed an approach that captures common practices of textbook formatting and organisation. 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 we start comparing formatting and structuring conventions across textbooks, we see very common patterns. Additionally, we use the explicit semantic relation between index terms and the organisational chapters to provide domain-specific knowledge and link the textbooks to an open knowledge base. As a result, our approach formalises the identified patterns and relations as rules and applies them according to a unified process to produced a textbook model that contains structural, content, and domain knowledge.

1.1. 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 (Déjean & Meunier, 2009; Gao et al., 2009).

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 analyse the structure of the textbooks. Other properties that facilitate the recognition of the TOC are (Déjean & Meunier, 2009): 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).

In the proposed approach, we have taken the TOC properties and generalised them into a set of rules that give insight into textbook structure and understanding. In our knowledge models, each textbook section represents a structural component annotated with its textual content and relations to other sections. Additionally, in our models, 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. Index

At the end of every well-designed textbook, is a manually created and curated index of important terms. Index creation is a tedious process (e.g. Wiley estimates that "adequate index preparation requires 10-15 hours per 100 typeset pages"[3]). This process follows elaborate guidelines stipulating index length and style, suggesting what can be good and bad candidates for index terms, advising on how to maintain consistency when creating hierarchical indices, etc. (Ament, 2001). Every index term is selected by a textbook author or a dedicated human indexer. A result is not just a collection of words, but, 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. And these pages do not simply mention index entries, but provide meaningful references by either introducing corresponding terms or elaborating them.

In the proposed approach, we have analysed the guidelines specified by several textbook publishers for creating indices, generalised them into a set of rules and implemented a rule-based component that extracts and formally represents term-based knowledge models and their annotations of textbook pages. At the very least, each extracted index is essentially a formally represented collection of manually selected labels for important notions in the domain and it comes with a manually annotated corpus of text, i.e. it becomes a machine-readable glossary.

1.3. Knowledge models

The extracted knowledge models using the TOC, Index, and other textbook components are not guaranteed to provide high-quality representation of a domain. They can potentially suffer from several drawbacks:

- subjectivity: they can contain terms that an author has used for explanatory purposes, but which are only marginally related to an objective picture of the domain semantics.
- coverage: on the other hand, they can miss important terms if a textbook does not cover (enough of) a particular part of a domain.
- granularity: although the authors are recommended to select index terms cohesively, sometimes, the terms in an index can be too detailed, too broad, or inconsistently vary in their granularity.
- lack of semantics: hierarchical indices can help specifying structural relations between sub- and super-entries, "see-also" cross-references can further facilitate linking related terms; however, unfortunately, most indices do not provide input even for these relations.

[3]https://authorservices.wiley.com/author-resources/book-authors/prepare-your-manuscript/indexing.html

To combat the potential problems of subjectivity, low coverage, poor granularity and lack of semantics, glossaries extracted from individual textbooks can be integrated with each other and linked to external models. In this study, we focus on one of the most popular such models— DBpedia (Bizer et al., 2009). It is a machine-understandable knowledge base automatically-extracted from Wikipedia. Currently, DBpedia contains 4.58 million resources 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.

1.4. Contribution

The contributions of this paper are threefold. Firstly, we present a general approach for first extracting knowledge models of textbooks, and then, enriching and connecting them with an open knowledge base. Secondly, we describe the implementation of the approach specifically for the PDF format and the DBpedia knowledge base. Finally, we present four different evaluations that show the accuracy of the approach, as well as the added value of the resulting enriched knowledge models.

The remainder of this paper is organised as follows. Section 2 presents related work. Section 3 provides the details of the proposed approach. Five evaluation experiments are discussed in Section 4. Finally, Section 5 outlines conclusions and future work.

2. Related work

2.1. Information extraction

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 organisation), to semantic (terminology and knowledge models).

2.1.1. Text and layout

Chao and Fan (2004) 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 recognised on the image and vector layers are saved separately. Hassan (2009) used a similar bottom-up approach to recognise 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.: pdftotext,[4] PDFBox,[5] GROBID,[6] and PdfAct.[7] Bast and Korzen (2017) 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, understating of reading order, and detection of word boundaries.

2.1.2. 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. (2012); Kern and Klampfl (2013) have developed several methods for detecting bibliography metadata by classifying text blocks and words within blocks. The CERMINE system uses both textual and geometric features to classify references (Tkaczyk et al., 2015). CiteSeerX relies on word-specific and line-specific features to recognise fifteen unique fields for metadata and citation extraction (Councill & Giles, 2008; J. Wu et al., 2015). Tables (Fang et al., 2011; Oro & Ruffolo, 2009; J. Wu et al., 2015), diagrams and figures (Gao et al., 2011; Shao & Futrelle, 2005), mathematical formulae (Baker et al., 2009; Lin et al., 2011), algorithms (Tuarob et al., 2016), and pseudo-code fragments (Tuarob et al., 2013, 2015) have been extracted using rule-based and machine learning techniques.

2.1.3. Document organisation

Tables of Contents (TOC) were often used as a source to recognise hierarchical organisation of documents. Gao et al. (2009) 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. (2012) and Z. Wu, Mitra, et al. (2013) extracted TOCs and bookmarks from documents with the help of rules and regular expressions. Marinai et al. (2010) presented a semi-automatic tool to recognise the TOC section based on identification of the TOC itself, identification of section headings, and connecting TOC entries to section headings across the book. For documents without a TOC, such as academic papers, other techniques have been proposed to extract their organisation. Hollingsworth et al. (2005) and Ramakrishnan et al. (2012) first harvested text blocks from research articles and then classified them into logical units based on templates and rules that characterise specific sections like title, abstract, introduction, and references. Tuarob et al. (2015) also used regular expressions to recognise 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.

[4] https://www.xpdfreader.com/pdftotext-man.html
[5] https://pdfbox.apache.org/
[6] https://github.com/kermitt2/grobid
[7] https://github.com/ad-freiburg/pdfact

2.1.4. Document terms

Index terms can be a source of document and domain-specific terms. Unlike TOCs, textbook indices surprisingly have not yet received much attention in the literature. One example known to us uses the index section to extract terminology of a single book by applying NLP tools and heuristic reasoning (Larrañaga et al., 2004). Terminology Extraction tasks are a different way to automatically identify and extract core vocabulary of a specialised domain in un- and semi-structured corpora. For example, Dwarakanath et al. (2013) presented a two-step method for the automatic extraction of glossary terms from software requirements documented in natural language. Lopes et al. (2010) used three different methods to extract compound terms from a text corpus of the domain of paediatrics.

2.1.5. Knowledge models

Research on extracting complete knowledge models from textbooks has been limited. Larrañaga et al. (2014) described a system that uses NLP techniques, heuristic reasoning, and ontologies for the semiautomatic extraction of a representation of the knowledge to be learned from electronic books. Wang et al. (2015) have extracted concept hierarchies from textbooks based on their TOCs and recognised DBpedia terms. Also, Sosnovsky et al. (2012) experimented with harvesting topic-based models from HTML textbooks using structures of their headings and mapping them into ontologies to facilitate more fine-grained personalisation of the textbook content. The work presented in this paper falls into this category— an approach implementing all stages of textbook knowledge extraction.

2.2. Information enrichment

Open Knowledge Bases, such as DBpedia, have been used in a range of different tasks (e.g. knowledge discovery, named entity recognition, word sense disambiguation) to discover and enrich information.

2.2.1. Knowledge discovery

DBpedia and other knowledge bases (KB) have been used before for exploratory knowledge search. Semantic Wonder Cloud is a graphical tool where the user selects an initial DBpedia resource, and then, the tool displays the ten most similar resources using a hybrid ranking algorithm (Mirizzi et al., 2010b). The user can continue the exploration by selecting one of the resources to get a new set of related resources. Discovery Hub 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 resources using a spreading activation algorithm and a sampling technique (Marie et al., 2013). Medelyan et al. (2008) used Wikipedia articles as topics and their titles as

controlled terms to discover topics in documents. Our enrichment strategy is not guided by one initial resource in particular, but instead, it uses a domain-specific glossary extracted from the index sections of textbooks to discover the resources in DBpedia that belong to the same target domain.

2.2.2. Linking resources to entities

The process of identifying the true sense 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 (Hahm et al., 2014). Word Sense Disambiguation (WSD) is the task of choosing the right sense or meaning for a word in a context using an exhaustive dictionary (Chang et al., 2016; Moro et al., 2014). Entity Linking (EL) and Named Entity Disambiguation (NED) are sometimes used indistinctly, where EL refers first to identifying entities and then linking them to a resource in a KB, and NED focuses on the potential ambiguity among several possible candidates in a KB (Chang et al., 2016). Typically, lexical ontologies, such as WordNet,[8] have been used in WSD (Agirre & Rigau, 1996; Navigli & Ponzetto, 2012), and global knowledge bases such as DBpedia in EL and NED (Kobilarov et al., 2009; Milne & Witten, 2008). Our enrichment process focuses on NED.

2.2.3. Named entity disambiguation

Different approaches and tools to solve NED tasks have been proposed. DBpedia Spotlight receives a text, detects the entities, and applies a Vector Space Model using context around the resources in DBpedia and the input text to disambiguate possible candidates for the entities (Mendes et al., 2011). Babelfly computes semantic signatures for concepts using Wikipedia, and then it links entities to the concepts based on a high-coherence densest subgraph algorithm (Moro et al., 2014). TAGME uses a collective agreement between a candidate entity and the possible candidates for other entities in the text for disambiguation (Ferragina & Scaiella, 2012). KORE uses keyphrase overlap relatedness to relate entities in the text to pre-extracted entities from Wikipedia (Hoffart et al., 2012). Other approaches use a notion of exclusivity-base relatedness to measure how similar two nodes are in a graph (Giannini et al., 2015), and a common subsumers algorithm between ambiguous entities and close entities that are unambiguous (Hulpus et al., 2015) for disambiguation.

Our disambiguation strategy differs from others in several aspects. First, we make use of the fact that textbooks belong to a specific domain, and we use a DBpedia category that matches the domain to create a core set of resources for context, if one is not provided. The DBpedia category is the only manual input data that our algorithm requires, in comparison to keywords (Rodríguez

[8] https://wordnet.princeton.edu/

Rocha et al., 2018) and seed entities (Mirizzi et al., 2010a) in other approaches. Second, the list of keywords from textbooks are in most cases abstract concepts and not concrete PLO (Person, Location, and Organisation) entities, so the use of individual categories or special formatting patterns for identifying those entities cannot be used (Bouarroudj & Boufaida, 2018; Demartini et al., 2013; Kobilarov et al., 2009). Finally, relying on prior probabilities to get the most popular (Han & Sun, 2011; Milne & Witten, 2008) or authoritative (Usbeck et al., 2014) entities is not useful in our case because we deal with domain-specific terms and not general entities.

2.2.4. 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 resources in the Semantic Web (Färber et al., 2015). Wikidata 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 (Vrandečić & Krötzsch, 2014). YAGO[9] has been developed since 2007. It has imported information from Wikipedia, WordNet, and GeoNames.[10] YAGO stores more than 10 million entities derived from about 120 million facts. Its latest version, YAGO3, is available for download. KBpedia[11] is another project combining knowledge from several public knowledge bases: Wikipedia, Wikidata, schema.org, DBpedia, GeoNames, OpenCyc,[12] and UMBEL.[13] 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 our approach uses DBpedia, it could be adapted to work with these knowledge bases.

3. Approach

Our approach first extracts a knowledge model from a textbook using its symbolic, structural and semantic elements. Then, the model is enriched with additional semantic information using DBpedia and the identified index terms in the textbook. Finally, the enriched knowledge model is serialised as an XML file using the Text Encoding Initiative (TEI). Figure 1 shows the overall workflow of the approach, including its three phases, seven main

[9] https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/downloads/
[10] http://www.geonames.org/
[11] http://kbpedia.org/
[12] https://www.cyc.com/opencyc/
[13] http://umbel.org/

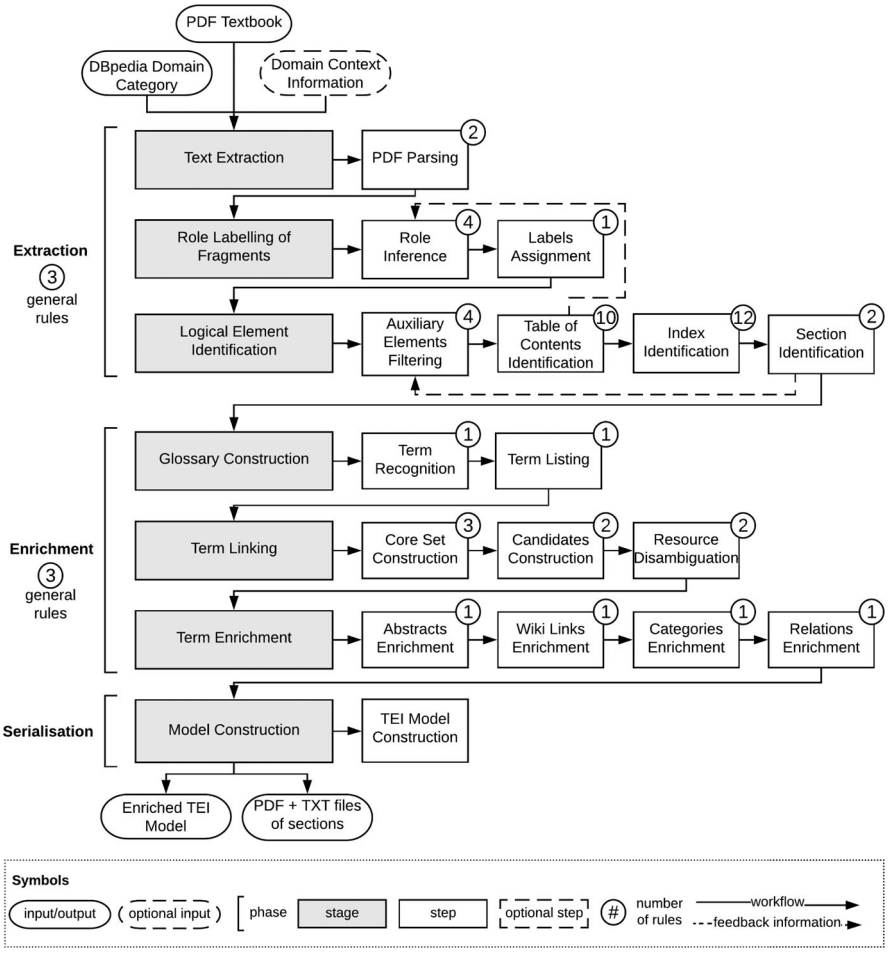

Figure 1. Phases, stages, and steps of the model extraction and enrichment workflow.

stages, steps at each stage, and the number of rules associated with each step. Altogether, 54 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, we reuse some ideas proposed in the literature for the creation of the rules. For example, we reconstruct text from the PDF in a bottom-up manner by merging page objects into more significant components similar to Hassan (2009). Roles of text fragments are inferred from formatting styles present in a textbook, which is related to Gao et al. (2011). To identify hierarchical structures of textbooks, we use a method similar to the one presented by Z. Wu, Mitra, et al. (2013) for TOC recognition and extraction. Also, we heavily utilise textbooks' index sections to extract fine-grained

domain terminology (Larrañaga et al., 2004). Our use of DBpedia categories to identify resources that belong to a domain is inspired by Mirizzi et al. (2010a) and Slabbekoorn et al. (2012). The use of extended context information for the disambiguation task is not new in the literature (Mendes et al., 2011; Navigli & Ponzetto, 2012). Many rules correspond directly to the application of guidelines defined in *The chicago manual of style* (2017). 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), and the structure of DBpedia (e.g. query a DBpedia resource). The modular nature of the rule-based approach support its gradual refinement. Each time we encounter a new variation of a formatting or structural pattern, we extend the approach by modifying an existing rule or adding a new one. To the best of our knowledge, this paper is the first attempt to propose a universal method for the automated extraction of knowledge models from regular textbooks.

3.1. Extraction

The first phase of our approach is the *extraction* of the knowledge model from the textbook. This phase has three main stages, seven steps, and 38 rules. As the first stage, our approach parses the textbook and retrieves its textual content in a structured way. At the next stage, text fragments are labelled to facilitate the recognition of different logical elements. Finally, all the different logical elements that are part of the textbook are recognised and added to the knowledge model of the textbook.

3.1.1. General rules

The first phase has three general rules. MULTICOLUMN_LAYOUT[14] uses the approach of Gao et al. (2008) 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 neighbouring zones. Empty areas with large left and right neighbour zones are marked as column margins. DEHYPHENATION follows the approach presented by Kern and Klampfl (2013) for the dehyphenation of words, which uses lists of hyphenation patterns taken from the TEX distribution[15] 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.

3.1.2. Text extraction

We define T as an input PDF textbook, where $T = \langle F, C \rangle$. F is the set of all formatting styles in T. Each style $f \in F$ has several attributes (font name, colour,

[14]Rule names are written in small caps and with underscores separating words.
[15]https://tug.org/tex-hyphen/

size, and extra features— bold/italic). C is the set of 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 organised according to basic textual elements (words, lines, and pages) using a bottom-up approach. We define w as a word that corresponds to a list $(c_1, \ldots, c_n) \mid \forall c \in C$ formed by characters. The list $W_x = (w_{1x}, w_{2x}, \ldots, w_{nx})$ corresponds to a line l_x formed by words. The list $L_y = (l_{1y}, l_{2y}, \ldots, l_{ny})$ corresponds to a page p_y formed by lines. Finally, the list $P = (p_1, p_2, \ldots, p_z)$ corresponds to all the pages that belong to T and it represents the textual content of the textbook.

3.1.2.1 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 pages (left to right and top to bottom), multiple problems have been detected when extracting subscripts, superscripts, formulae, and rotated text. Therefore, we use two additional rules to process these cases. 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 set 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 recognised, and hyphenated words are not yet merged, because PDFBox does not support such features. Our approach handles such cases in the following stages. At the end of this first stage, a simplified textbook model M of T is created: $M = \langle P \rangle$.

3.1.3. Role labelling 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.

3.1.3.1 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 recognised 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 colour 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: $f \mid o(f)$ 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. We are extending this rule to recognise 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 (see Section 3.1.4.2). 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.

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

3.1.4. Logical element identification

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

3.1.4.1 Auxiliary elements filtering. First, REPEATED_LINES is used to detect (and then remove) header and footer lines across pages. A sample of pages $P_s = \{p_a, p_b, \ldots, p_m\} \mid P_s \subset P$ 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 (see Section 3.1.4.4).

The final type of auxiliary page elements that we process 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, $M = \langle P, N \rangle$.

3.1.4.2 Table of Contents identification. The TOC section provides organisational 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 (Z. Wu, Das, et al., 2013; Z. Wu, Mitra, et al., 2013) used the first 20 pages of the book for this step, but we found textbooks 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 heading 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 $TOC = \{e_0, e_1, \ldots, e_n\}$.

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 (NER) 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 Z. Wu, Mitra, et al. (2013), 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:

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

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, $\forall e \in TOC$:

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

TOC entries contains 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, $M = \langle P, N, TOC \rangle$. Figure 2(a) shows different elements in the TOC that are recognised by the rules.

3.1.4.3 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 the creation of 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 recognised 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

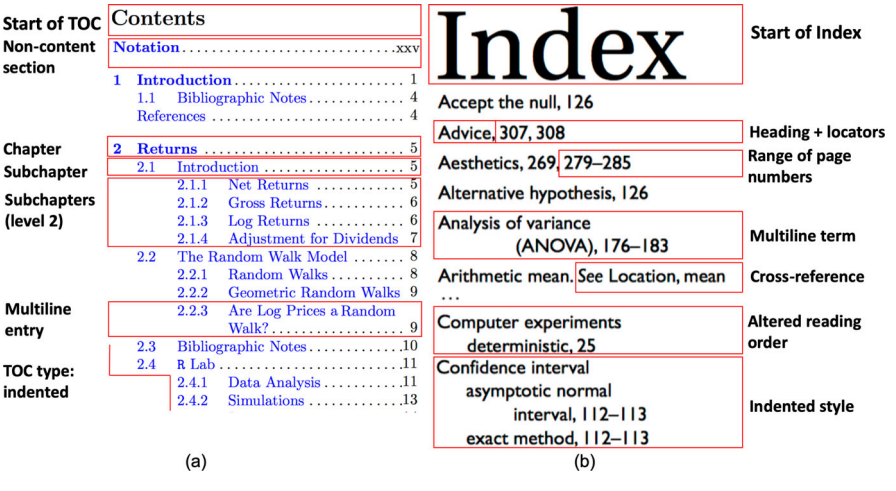

Figure 2. Examples of elements identified in (a) TOC and (b) Index sections.

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 leftmost 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 alphabetisation 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 textbook,[16] 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 end-notes (e.g. 25N3). Each type is first identified using a dedicated regular expression, and then normalised 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.

[16]Name of a person, a place, an object, or an abstraction.

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}, ...) if $s(l_{x+n}) > s(l_x)$. Sometimes if a nested index term starts in a page and continues in 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, $M = \langle P, N, TOC, I \rangle$. Figure 2(b) shows the different elements in the Index section that are recognised using the rules.

3.1.4.4 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 for easy sharing of specific parts of the textbook, 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, $M = \langle P, N, TOC, I, PDF, TXT \rangle$.

3.2. Enrichment

The second phase of our approach is the *enrichment* of the extracted knowledge model with semantic information using the open knowledge base DBpedia. This phase has three main stages, nine steps, and 16 rules. As the first stage, our approach formally represents all the index terms as a machine-readable glossary linked to pages across the textbook. At the next stage, we extract DBpedia resources that belong to the target domain of the textbook and link them with relevant concepts from the extracted glossary. Lastly, 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. We also argue that adding more textbooks from the same domain should result not only in a joint hyperspace of cross-linked textbooks, but also in a more complete and objective model.

3.2.1. General rules

The current phase has three general rules. DBpedia_Resource is used to query a term in DBpedia and get its URI if it exists. The rule applies the convention for resources 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 correspond to a subject resource in DBpedia, then a matching resource has been identified.

Final_Resource says that if a resource has the DBpedia redirect property (`dbo:wikiPageRedirects`[17]), the object value of the property is the final URI of the resource. Redirects are used in DBpedia to indicate the final URI of a resource and include synonyms, common misspellings, and acronyms. For example, the resource *dbr:Gaussian_distribution* is redirected to *dbr:Normal_distribution*.

The last general rule, Ambiguous_Resources, is used to create a set of resources related to an ambiguous resource. A resource is ambiguous if one

[17]The list of the namespace prefixes used in this paper can be found at https://dbpedia.org/sparql?help=nsdecl

of two possible cases happens. Case ONE is when the found resource has the `dbo:wikiPageDisambiguates` property. For example, the *dbr:Distribution* resource links to 33 resources using the `dbo:wikiPageDisambiguates` property. Case TWO is when the resource does not have the `dbo:wikiPageDisambiguates` property, but it is listed as an object with other entities in a resource that uses the `dbo:wikiPageDisambiguates` property. For example, the *dbr: Median_lethal_dose* resource appears in the *dbr:MLD* resource through the mentioned property. If either case applies, the rule generates a set with all the listed resources using the disambiguation property.

Several rules use two functions. $q(\cdot)$ 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 $r(\cdot)$ returns the URI of a resource associated with a glossary term. The returned URI corresponds to the final resource after any redirect has been followed. In the case that a term has multiple candidate resources, the function returns the URI of the chosen resource after the disambiguation step.

3.2.2. Glossary construction

The first stage of the enrichment phase is the construction of a glossary that contains all the index terms in the textbook model *M*.

3.2.2.1 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 standardised reading order, the rule READING_LABEL identifies the right reading label (r) of each index term ($i \in I$). For example, the hierarchical index entry *distribution : (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 right reading labels. 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 our rule is able to detect such an occurrence. READING LABEL uses a term recognition algorithm to check the possible labels (*LABELS*) against the reference text in the form of sentences (*SENTENCES*) of each term (according to the locators of the index entry) to identify the right labels. *LABELS* 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."the lavish green grass"). 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 (c and c') 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 is marked as the right reading label (r) for the index term:

$$\text{if } \exists lab \in LABELS \text{ and } \exists sen \in SENTENCES$$
$$\text{and } \forall c \in lab\ \exists c' \in sen : c(c, c'), r \mapsto lab.$$

3.2.2.2 Term listing. In this step, the index entries and the recognised reading labels are used to construct a glossary of terms G. For each index term, a glossary term g_x is added to G. Each glossary term corresponds to a list ($property_1, \ldots, property_n$) formed by properties. Initially, a glossary term contains the main label ($main$) and a set of alternative labels (ALT). The rule TERM_LABELS is used to list for each term $i \in I$, its different labels, formally:

$$\text{if } \exists r \text{ for } i,\ main \mapsto r \text{ and } ALT \mapsto \emptyset;$$
$$\text{otherwise, } main \mapsto natural(i) \text{ and } ALT \to permu(i) - natural(i).$$

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 $i_0 =$ "adjacent value : lower," $natural(i_0) =$ "adjacent value lower" and $permu(i_0) =$ { "adjacent value lower," "lower adjacent value"}.

Sometimes authors index the same term in several ways. For example, the term "normal distribution" sometimes appear as two index entries: "normal distribution" and "distribution : 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 T is created. Initially, $E = \langle G \rangle$.

3.2.3. Term linking

In this stage, the terms from the constructed glossary G are linked to DBpedia resources. Finding the right resource 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 core set of terms that only have one possible matching resource in DBpedia is constructed from the glossary. Then, the terms that have multiple candidate resources 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 resource from the list of candidates. If no initial context information is provided, the abstracts in DBpedia from the resources in the core set are used as context. This stage has three steps.

3.2.3.1 Core set construction. In this stage, a core set of terms *CORE* is created. This set includes the terms from G for which only a single DBpedia resource has been found, which is unambiguously the right resource for the term. The resources are found with the help of the DBpedia categories. Categories[18] correspond to a special type of resources in DBpedia used to classify and group together resources on similar subjects. They are ordered in a broad and non-strict hierarchy (i.e. sub-categories can have multiple super-categories). We claim that if a term has only one possible candidate resource in DBpedia and that resource belongs to a (sub-)category that is a part of the target DBPedia domain; it is safe to link the term to such resource unambiguously.

Inspired by Mirizzi et al. (2010a) and Slabbekoorn et al. (2012), we use the main DBpedia domain category given as input (cat_{main}) 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 resources for the core 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:

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

[18] https://en.wikipedia.org/wiki/Help:Category, https://wiki.dbpedia.org/services-resources/datasets/dbpedia-datasets#h434-7

After the set of in-domain categories has been constructed, each $g \in G$ is checked to see if it belongs to the core set using the three general rules of this phase and an additional rule. First, DBpedia_Resource is used to query DBpedia and get the corresponding resource. Then, if a resource was found, the rule Final_Resource is used to get the final URI of the resource. After that, the set of resources related to the found resource (*AMB*) is retrieved using the rule Ambiguous_Resources. Finally, the rule Core_Set_Resource uses the `dct:subject` property to get the categories of the resource and decide if the term belongs to *CORE*:

$$\text{if } |AMB| = 0 \text{ and } \exists cat \in q(r(g), \text{dct:subject}, ?) : cat \in CAT, g \in CORE;$$

$$\text{otherwise, } g \in CORE.$$

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 *CORE* forms the domain context information *DOMAIN*. Abstracts are retrieved using the `dbo:abstract` property and the $q(\cdot)$ function.

Now, $E = \langle G, CAT, CORE, DOMAIN \rangle$.

3.2.3.2 Candidates construction. After the core set has been created, the remaining terms in the glossary still need to be linked to resources in DBpedia. In this step, a list of candidate resources to be associate to each of those remaining terms is constructed. We define *cand* as a candidate resource that is associate to a $g_x \notin CORE$. The set $CAND_x = \{cand_{1x}, cand_{2x}, \ldots, cand_{nx}\}$ corresponds to a list of candidates $list_x$ formed by candidate resources. The list $LIST = (list_1, list_2, \ldots, list_n)$ corresponds to all the candidate lists from all the glossary terms that do not belong to *CORE*. Each candidate *cand* has the URI of the resource *uri* and its context information *ctx*.

For each $g_x \notin CORE$, the three general rules are used to find a matching DBpedia resource and get the set of related resources *AMB* if the term is ambiguous. One example of an ambiguous term is "mean." DBpedia returns the *dbr:Mean* resource when querying the term. This resource does not have the `dbo:wikiPageDisambiguates` property, but it appears in a group with other resources in the *dbr:Mean_(disambiguation)* resource using the `dbo:wikiPageDisambiguates` property.[19] Figure 3 shows the list of related resources for the term "mean" after resolving the disambiguation property using the rule Ambiguous_Resources.

Then, the list of candidates is created using Candidate_List. This rule says that the matching DBpedia resource and all the resources from the set of related

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

- 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 3. Candidate list for the term "mean."

resources are the candidates of the term, which are added to $CAND_x$. When each candidate resource *cand* is created, its context information is retrieved to be used in the disambiguation process. Since our disambiguation stage uses cosine similarity, the primary context information for a resource is its abstract. Additionally, based on other approaches (Mendes et al., 2011; Navigli & Ponzetto, 2012) that gather context information such as all paragraphs mentioning a candidate resource in Wikipedia or the titles of other resources 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 resources 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:

$$LINKS = q(?, \text{dbo:wikiPageWikiLink}, cand);$$

$$ctx = q(cand, \text{dbo:abstract}, ?) \cup \{q(link, \text{dbo:abstract}, ?) : link \in LINKS\}.$$

We assume that the candidate resources that belong to the same domain as the term g_x will get abstracts also from other resources 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, $E = \langle G, CAT, CORE, DOMAIN, LIST \rangle$.

3.2.3.3 Resource disambiguation. The final step of the current stage is to apply a disambiguation strategy to choose the best resource 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 resource with the highest similarity score that surpasses the minimum threshold. The selection of the

resources is incrementally to first select the resources that are more likely to be the right match.

SIMILARITY_SCORE computes a similarity score *sim* for each candidate resource *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 were 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_x, 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 resources 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 resources are retrieved: *dbr:Method_of_moments_(statistics)* and *dbr:Method_of_moments_(probability_theory)*. In our example, the *dbr:Method_of_moments_(statistics)* resource gets a similarity value of 1. Case TWO assigns $sim = 0.9 + (sim/10)$ to a *cand* if only it has the same name as its term g_x (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 $sim = sim + 20\%$ to a *cand* if it is the resource returned when querying DBpedia with the rule DBPEDIA_RESOURCE, and it is not a disambiguation page. In our "mean" example, from the candidate list of resources (see Figure 3) the similarity score of the *dbr:Mean* resource 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_RESOURCES chooses the right resources from the candidates in an incremental process. First, the rules uses 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 *ctc*'s of the chosen resources 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 resources that get a higher similarity score, *DOMAIN* is expanded with the context information from the selected resources. 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 resource for each term.

At the end of this step, the terms from the glossary with an identified matching DBpedia resource are saved as the list of selected resources *SLCT*. Now, $E = \langle G, CAT, CORE, DOMAIN, LIST, SLCT \rangle$.

3.2.4. Term enrichment

The final stage of the *enrichment* phase is to use the discovered DBpedia resources 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 $g_x \in (CORE \cup SLCT)$.

3.2.4.1 Abstracts enrichment. The rule ABSTRACT adds the abstract from its matching DBpedia resource to each term: $g_x = g_x \cup q(r(g_x)$, dbo:abstract, ?). Abstracts bring summarised definitions for the index terms of the textbooks.

3.2.4.2 Wikipedia links enrichment. The Wikipedia pages from where the information was extracted to create the resources in DBpedia are used to enrich the terms further. The rule WIKIPEDIA_LINK uses the the `foaf:primaryTopic` property of the resources to enrich each term with the link from its Wikipedia page: $g_x = g_x \cup q(r(g_x)$, 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).

3.2.4.3 Categories enrichment. The `dct:subject` property is used by the rule CATEGORIES to enrich each term with its DBpedia categories: $g_x = g_x \cup q(r(g_x)$, 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.

3.2.4.4 Term relations enrichment. Finally, we exploit the information about linked Wikipedia pages stored in DBpedia to discover relations among the terms in the glossary. The general idea is that if two resources in DBpedia are linked together, the corresponding terms in the glossary are also related. For each term with a linked resource, the rule RELATIONS queries DBpedia uses the *dbo:wikiPageWikiLink* property to retrieve all other entities that the current resource links to in its corresponding Wikipedia page. Then, for each retrieved resource that is linked to a term, a relation is created between the two terms. Formally:

$$LINKS = q(r(g_x), \text{dbo:wikiPageWikiLink}, ?);$$

$$REL_x = \{g_y : g_y \in G, r(g_y) \in LINKS\};$$

$$g_x = g_x \cup REL_x.$$

For example, the term "mean" is linked to the resource *dbr:Mean*, which links

to 87 resources. One of those resources is *dbr:Mode_(statistics)*. If in the glossary the term "mode" is linked to *dbr:Mode_(statistics)*, then a relation between "mean" and "mode" is created.

For the moment, we only specify that two terms are related in a general manner without detailing the type of relationship. In the future we plan to exploit both the hierarchical structure of index terms and the content in textbooks to be able to discover subtopic hierarchies (Bayomi & Lawless, 2018) and more specific relations (e.g. *is-a* relations) (Larrañaga et al., 2004).

3.3. Serialisation

The last phase of our approach is the *serialisation* of the enriched knowledge model. This phase has only one stage, one step, and no rules.

3.3.1. Model construction

After all the information has been extracted and terms have been enriched, the final stage is to map all the components of the knowledge model to TEI elements, and then save the created TEI model as an XML file.

3.3.1.1 TEI model construction. We use the Text Encoding Initiative (TEI)[20] 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. We use the P5 Guidelines to express the information collected in model *M*, and RDF relationships to incorporate the additional semantic information in model *E*. Since the TEI Guidelines do not provide a mechanism to integrate RDF with TEI,[21] we follow some of the previous approaches (Ruiz Fabo et al., 2018; Tittel et al., 2018) and add RDFa[22] attributes to the TEI attribute class `att.global.linking`: `@about`, `@property` and `@resource`. Those attributes allow us to represent RDF tuples: `@about` for the subject, `@property` predicate and `@resource` object.

In this step, the elements from *M* and *E* are mapped to TEI elements. Our TEI model groups all the information into three categories that represent the information extracted from the textbook: Structure, Content, and Domain Knowledge. Table 1 presents the mapping between the extracted information and the used TEI elements. Our choices for the TEI elements share similarities with a previous proposal of TEI elements for textbook research (Stahn et al., 2016).

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

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

[22] http://rdfa.info/

Table 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*)	(text division)	@type="contents"	
	TOC entries ($e_x \in TOC$)	<list> + <item> + <ref> (reference)	<ref @target="segment id">	Content of <ref> is the page number, target points to its content section
Content	Part, chapter, or subchapter (*TOC*+ *TXT*)	(text division)	@type="part|chapter|section" + @id="[segment id]"	
	Title of part, chapter, or subchapter ($s_x \in TXT$)	<head> (heading)		Heading/subheading
	Pages ($p_z \in P$) + page numbers (N)	<pb> (page beginning)	@n="[page number]"	
	Fragments ($t_y \in T_z$)	<ab> (anonymous block)		Body text fragments
	Lines ($l_x \in L_y$)	<lb> (line beginning)		
	Words ($w_w \in W_x$)	<w> (word)		
Domain Knowledge	Index section (I)	(text division)	@type="index"	
	Index terms ($i_x \in I$)	<list> + <item> + <ref> (reference)	<ref @target="segment id">	Content of <ref> is a page reference, target points to its content section
	Semantic information ($g_x \in G$)	<seg> (arbitrary segment) + <gloss> (gloss or definition) + <label> + <ref> (reference)	@about="[URI]" + @property="[namespace:property]" + @resource="[URI]"	<seg> groups the info for a term: <gloss> is used for abstracts, <label> for reading label, <ref> for Wikipedia pages, categories, and term relations

4. Evaluation

Four evaluation experiments have been conducted. First, we have tested the quality of the outcomes produced by the *extraction* phase of the approach that are used to construct the textbook model. Secondly, we have examined how the obtained model, without any enrichment, can support one of its possible knowledge-driven applications: linking relevant sections across textbooks within the same domain. In the third evaluation, we have tested the quality of the outcomes produced by the *enrichment* phase of the approach that are used to enrich the textbook model. For the last evaluation, we were interested to examine how adding more textbooks would affect the ability of the approach to connect DBpedia resources from a target domain.

4.1. *Accuracy of the* extraction *phase*

The goal of the first experiment has been to find out if the first phase of 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, we have verified the information extracted from the PDF versions of textbooks 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, we use it as the ground truth. We hypothesise that if the information obtained from the two versions of a textbook matches, that means the approach processes PDF correctly. First, we have compared the results of the approach against two popular PDF processing tools to demonstrate the added value of the rule-based workflow when extracting raw text. In addition, we verified how well the approach elicits structural information focusing on TOC entries and index terms.

4.1.1. Procedure

We have used SpringerLink[23] as the content source 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 in other domains, we have added to the test set five textbooks per each of the following domains: *computer science*, *history*, and *literature*.[24]

[23] https://link.springer.com/

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

This experiment has focused on three different outcomes from the first three stages of the approach (the last stage is the extraction of the textbook knowledge model itself). To evaluate the quality of text extraction stage, we compared its results with the results of the 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 our 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 our approach. Additionally, in the mentioned evaluation of PDF text extractors (Bast & Korzen, 2017), it yielded satisfactory results on all evaluation criteria. PDF versions of the entire corpus of textbooks from the four domains were processed by all three tools. *Google's Diff Match Patch library*[25] was used to synchronise and compare the extracted texts with the ground truth generated using the EPUB versions of the textbooks. Table 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 (Doucet et al., 2013, 2011), each entry is classified as correct if the right heading, hierarchy level, and page number match the ground truth. Standard precision and recall are reported over the total number of evaluated elements.

Finally, we have evaluated the extracted domain terms 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). For this evaluation, we have also used standard precision and recall over the total number of evaluated elements as metrics.

Table 3 shows the results for the structural and domain terms evaluations.

4.1.2. Analysis

The results of the text extraction evaluation show that our rule-based system demonstrates the best accuracy among the three evaluated tools, followed by PDFBox and then PdfAct. The results do not reach the 100% mark because of 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

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

Table 2. Text extraction.

		Our approach		PDFBox		PdfAct	
Domain	Chars EPUB (#)	Chars (#)	Matched (%)	Chars (#)	Matched (%)	Chars (#)	Matched (%)
Statistics	641,697	730,060	82.09	737,015	79.93	727,502	74.04
Computer Science	373,615	374,985	96.61	410,238	85.42	407,373	81.34
History	662,300	660,493	98.07	671,693	96.50	661,503	91.51
Literature	571,277	568,561	98.64	578,121	97.03	565,463	89.89

Table 3. Extraction of TOC entries and domain terms.

	TOC			Index		
Domain	No.	*P* (%)	*R* (%)	No.	*P* (%)	*R* (%)
Statistics	7968	99.70	99.70	19,779	99.78	99.80
Computer Science	811	100	100	1661	97.77	98.00
History	216	100	100	2567	97.78	96.66
Literature	113	100	100	1800	98.94	98.07

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 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 our 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 well as the two other methods.

When it comes to the identification of structural elements, precision and recall values are high across all domains. Results for the detection of TOC entries show that the number of recognised 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, we observed that some entries were misplaced in the TOC hierarchy (e.g. third-level chapter instead of second-level) because of the wrongly recognised mathematical characters (e.g. $\surd$). 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. Several terms have been recognised incorrectly because of special cases that our 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 701–705). A few detected problems were due to the dehyphenation of

words (e.g. *t-* was merged with *distribution* as *tdistribution*). 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.

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, we have use the knowledge models of two textbooks to cross-link relevant sections among them. Additionally, we have compared the results from our linking model to two baselines. We believed that the use of the extracted domain knowledge of the textbooks will produce accurate matches between the sections of textbooks.

4.2.1. Procedure

We have followed a procedure similar to the one presented by Guerra et al. (2013). Two introductory statistics textbooks have been used: BOOK#1 (Walpole et al., 2012) and BOOK#2 (Devore & Berk, 2012). 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 Glossary that contains more than 3500 statistical terms and their synonyms.[26] 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 Vector Space Model (VSM) (Salton et al., 1975) of all (sub)chapters and (sub)sections of the 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 have been used to compute similarity between all sections of BOOK#1, and sections of BOOK#2.

To evaluate the effectiveness of our linking model, we have used average NDCG@1, @3, and @5 scores. NDCG (normalized discounted cumulative gain) (Järvelin & Kekäläinen, 2002) 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.

[26] http://isi.cbs.nl/glossary/

We have used the linking produced by the experts as the ground truth for the NDCG measures. Additionally, we have used two baselines for comparison: the standard *TFIDF* model (Salton & Buckley, 1988) and the *LDA* model (Blei et al., 2003) built based on BOOK#1 using 2000 iterations, 158 topics (the number of sections in BOOK#1), and $\alpha = 50$. We have used the Hellinger distance as the similarity measure for LDA. Both baselines have used the textual content of each part of the textbooks with basic pre-processing (lowercase, stop-words, and stemming). Additionally, we have computed an ensemble model 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 4.

4.2.2. 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 (Guerra et al., 2013): TFIDF model has the lowest accuracy, followed by LDA, followed by the ensemble model. The difference between our 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 our 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. The obtained results support our hypothesis that the domain knowledge information extracted using our approach is useful to cross-link sections of textbooks.

We have conducted a range of pairwise *t*-test to check if our model significantly outperforms the baselines across the BOOK#1 sections. Table 4 presents the results including Bonferroni-adjusted p-values.

4.3. Accuracy of the enrichment *phase*

Our goal for this evaluation was to find out if the second phase of the approach can recognise the index terms in the pages of the textbooks and link them to the right DBpedia resources. In this evaluation, we have compared the number of recognised index terms and reference pages from our approach against a baseline. We have hypothesised that the use of Part-of-speech tagging to recognise the nouns in the index terms would increase the number of recognised index terms in the pages of the textbook in comparison of textual search. Additionally, we have compared the number of correctly linked resources using our

approach against two baselines. We believed that the construction of a core set of resources, and the use of the context information extracted from the resources in that set will perform better at disambiguating and choosing the right resource for each term than approaches that do not use context information.

4.3.1. Procedure

For this evaluation we have used three textbooks. The first two in the statistics domain: STAT#1 (Dekking et al., 2005), and STAT#2 (Walpole et al., 2012). The third book is for information retrieval: IR#1 (Manning et al., 2009). This and the final experiment were conducted using a local copy of the DBpedia version 2016-10.[27]

First, for each of the textbooks, the index terms and the reference pages (locators) have been extracted from their knowledge models. Then, we have applied to the index terms and reference pages our term recognition algorithm (*TRA*) and a baseline strategy for comparison. The *baseline (BL)* tried all possible labels of each index term using simple textual search to find the index term in the reference pages. Table 5 shows the results for the term recognition evaluation. We report the percentage of index terms where the label (the right reading order) is recognised in at least one reference page (% index terms) and the percentage of reference pages where the index term is recognised (% reference pages).

In the second part of this evaluation, we have tested the precision and accuracy of the second stage in the enrichment phase, the linking of the index terms to DBpedia. For STAT#1 and STAT#2, we ran our approach given as input the *dbc:Statistics* category 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 core set. After the "Core Set Construction" and "Candidates Construction" steps, there were 47 terms in the core 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 resource disambiguation step, we used our strategy and two baselines that do not use context information for disambiguation. Based on Mendes et al. (2011), we used the following baselines:

- *Random Baseline (BL1)* selected randomly one of the resources in the candidates list as the right resource. This baseline will show if our algorithm selects the right resource better than chance.
- *Default Sense Baseline (BL2)* selected the resource in the candidates list which is the most referenced resource in Wikipedia from other pages. This baseline

[27] https://wiki.dbpedia.org/develop/datasets/dbpedia-version-2016-10

Table 4. Semantic linking of textbook sections.

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

Table 5. Term recognition.

			BL		TRA	
Textbook	No. index terms	No. reference pages	% index terms	% reference pages	% index terms	% reference pages
STAT#1	651	738	63.44	62.60	82.64	81.44
STAT#2	591	859	82.74	83.70	93.06	92.89
IR#1	608	774	87.66	85.27	94.08	92.64

will show the impact of using a specific domain for disambiguation in contrast to choosing just the most used resource in the candidates list.

For precision we used the number of *correctly selected* resources from the candidates lists divided by the number of *selected* resources from the candidates lists. Recall was computed dividing the number of *correctly selected* resources by the number of *relevant* resources. For evaluating the selected resources, we manually marked from each of the candidates lists the right resource. The total number of relevant items correspond to the number of terms that belong to the domain of the textbook and had a matching resource in their

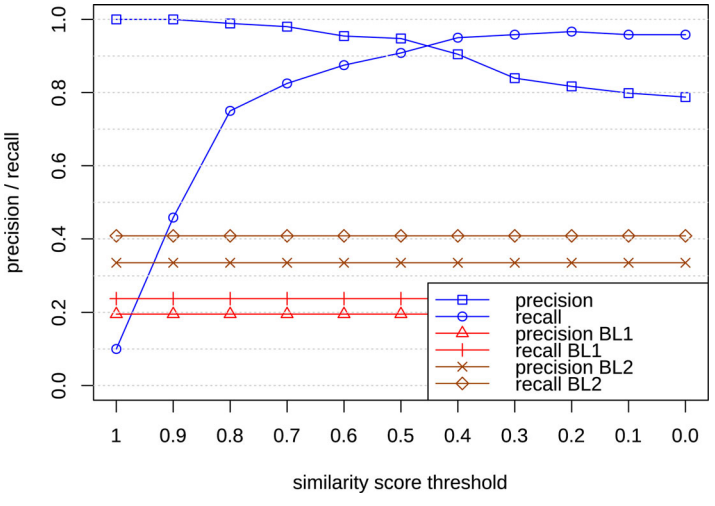

Figure 4. Precision and recall for the linking evaluation of STAT#1 textbook.

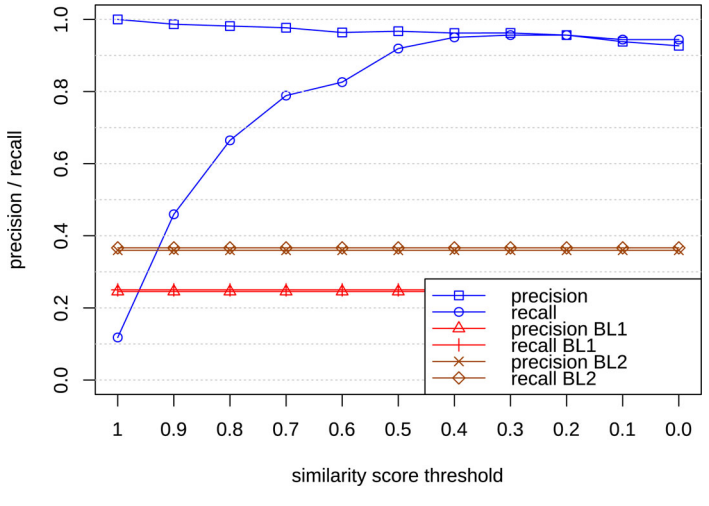

Figure 5. Precision and recall for the linking evaluation of STAT#2 textbook.

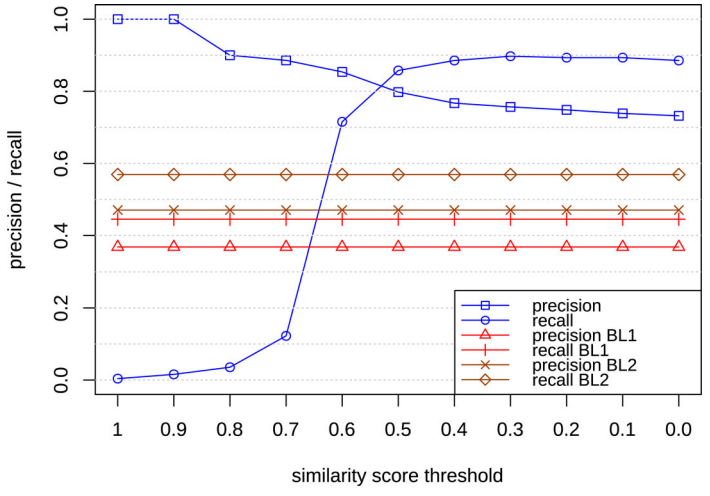

Figure 6. Precision and recall for the linking evaluation of IR#1 textbook.

candidates lists. Since our strategy uses a minimum threshold for selecting a candidate as the right resource, we also tested which value would give the best results. Figures 4–6 show for each textbook the precision and recall of our algorithm 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.

4.3.2. Analysis

Results show that the performance of our 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 in 19.20% the number of recognised index terms and in 18.84% the number of pages where the index terms are recognised to the total number. The algorithm achieves the smallest increases in the IR#1 textbook, 6.42% and 7.37% respectively. Our algorithm can effectively detect index terms that appear fragmented in the text pages (see the "Bernoulli" example in Section 3.2.2.1), but its performance decreases recognising 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 our algorithm cannot recognise 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. We plan to add an external model (e.g. a dictionary) to deal with synonyms, and increase the flexibility of the algorithm to recognise different parts of the index terms.

When it comes to the right linking of terms to DBpedia resources, 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, our strategy obtains the best results. The use of context information achieves better disambiguation than just selecting the most used resources because the terms of the glossary belong to the same domain, and our strategy exploits this characteristic by constructing the core set of resources. 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, we can favour precision or recall depending on the final use of the algorithm. In most use cases, we prefer higher precision (given a reasonable recall) to minimise the number of errors in the model. In general, as the threshold decreases and gets closer to 0, the algorithm selects not only incorrect resources, but also irrelevant ones; they do not belong to any of the related domains of the textbooks.

We also carried out three evaluations given specific context information and not using the one constructed from the core set. The context information was obtained using the textbooks. First, for each index term, we extracted the sentences in which the term appears. 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 core 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 core set is as useful for the disambiguation algorithm as the context directly taken from the textbooks.

Overall, this experiment has demonstrated that the proposed enrichment phase can identify the index terms in the pages of the textbook and link them to the right resources with high precision and recall.

4.4. Discovery of resources

The goal of the final evaluation experiment was to determine the effect of adding more textbooks in the glossary construction step and test how many resources the approach will find in DBpedia for a target domain. In this evaluation, we used a ground truth of DBpedia resources in the statistics domain to

quantify the resources in the domain that can be discovered. We believed that by integrating glossaries from multiple textbooks in the same domain we can 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 for us the subset of DBpedia resources that belong to the same domain as the textbooks.

4.4.1. Procedure

To test our hypothesis, we have used ten introductory statistics textbooks (Dalgaard, 2011; Dekking et al., 2005; Devore & Berk, 2012; Faber, 2012; Finkelstein, 2009; Kaltenbach, 2012; Madsen, 2011; Shanmugam & Chattamvelli, 2015; Ubøe, 2017; Walpole et al., 2012). We wanted to find out how many of the linked resources belong to the statistics domain, and compared that to the total number of resources that belong to the domain. For this evaluation, we defined precision as the number of *linked resources that belong to the domain* divided by the number of *linked resources*. The recall was the number of *unique linked resources that belong to the domain* divided by the *total number of resources in DBpedia that belong to the domain*. To determine the actual resources in DBpedia that belong to the statistics domain we constructed a ground truth using the ISI Multilingual Glossary of Statistical Terms.[28] The ISI Glossary translates more than 3500 statistical terms into 31 languages. It was created by The International Statistical Institute which recognised the need for an affordable multilingual glossary of statistical terms. We used our approach to link the terms from the comprehensive ISI Glossary to DBpedia resources, and after manually checking the obtained linked terms, we got a ground truth of 1049 resources in the statistics domain.

We evaluated three configurations using the ten textbooks. Figure 7 presents the results for resource discovery using individual textbooks; the results are averaged across all ten textbooks to account for potential differences between them. Figure 8 shows the average discovery of resources of two sets of five textbooks each. Finally, Figure 9 shows the discovery of resources when using all of the ten textbooks combined.

4.4.2. 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 7). 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 resources in DBpedia (average of 132). The second configuration of five

[28] http://isi.cbs.nl/glossary/

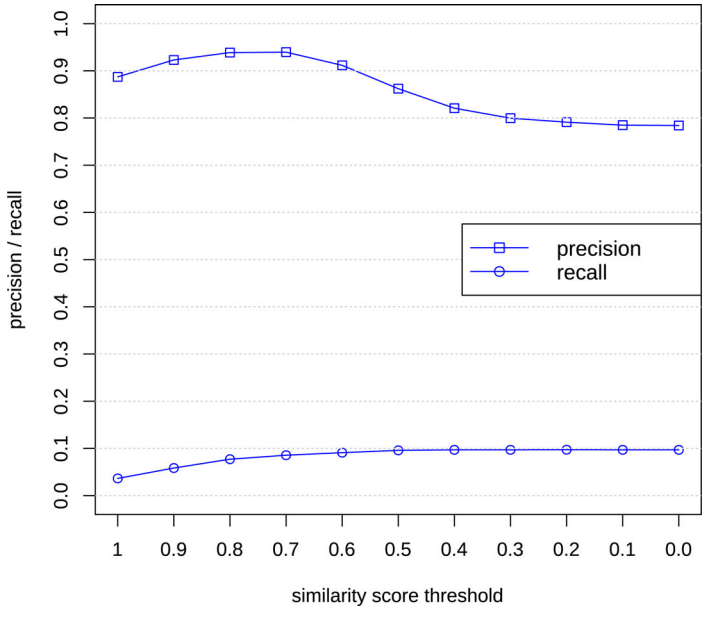

Figure 7. Precision and recall for resource discovery using one textbook (averaged over ten books).

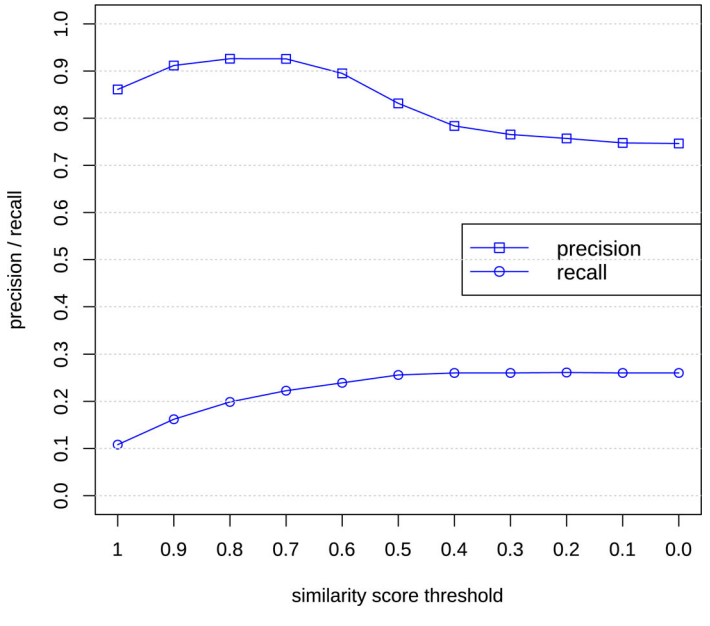

Figure 8. Precision and recall for resource discovery using five textbooks (averaged over two sets of five books).

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 8, as the number of terms increases, the recall goes up, reaching up to 0.261, but the precision goes

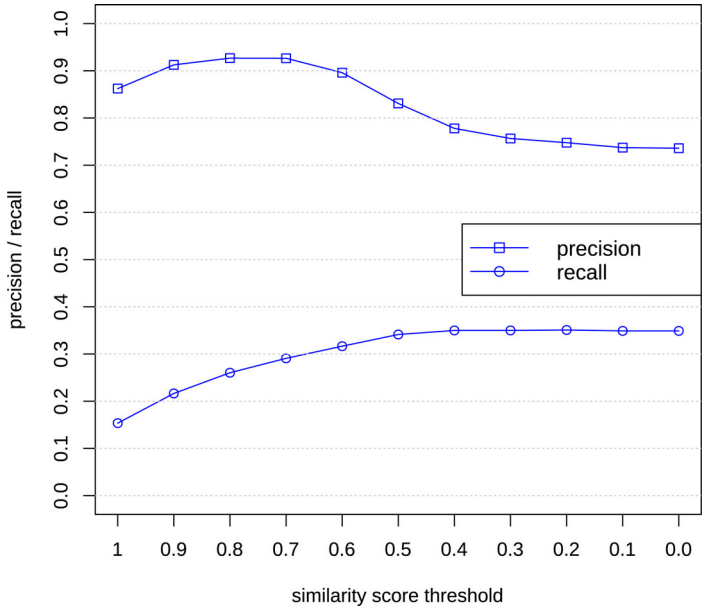

Figure 9. Precision and recall for resource discovery using ten textbooks.

down to the minimum value of 0.746. The final configuration (Figure 9) which includes ten textbooks and 4455 terms (854 with candidate resources) gets a recall of 0.350, which means that more than a third of all resources 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 resources in the domain, additional textbooks from a more diverse and applied side of statistics should be used. For example, the term "cox's theorem"[29] is a statistical theorem, it is not part of any of the ten textbooks, but it belongs to our target domain. Using Google Books, this term was found in three textbooks with particular topics: uncertainty theory (Liu, 2007), statistical evidence measurement (Evans, 2015), and universal artificial intelligence (Hutter, 2010). 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 recalls increments 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 resources *dbr:Slope*, *dbr:Law_(stochastic_processes)*, and *dbr:Logarithmic_scale* are linked to terms in the configurations, but those resources do not correspond to terms from the ISI glossary; therefore, they are not part of the ground truth. Filtering

[29]http://dbpedia.org/resource/Cox's_theorem

linked resources that do not belong to the domain could be accomplished by further exploiting the categories of the resources or by analysing the graph structure of DBpedia to detect resources that do not belong to the cluster of in-domain resources. The obtained recall values support our hypothesis that it is possible to discover all resources in a domain using textbooks as the source of concepts in a domain.

5. Conclusions and future work

This paper presents the implementation details and the evaluation of a novel approach for automated extraction and enrichment of knowledge models from textbooks. Results of the evaluation experiments show: (1) the proposed extraction phase of the approach is capable of processing PDF textbooks with high accuracy; (2) the added value of the extracted knowledge models by using them to link sections across textbooks within the same domain; (3) the proposed enrichment phase of the approach links terms to the right external resources with high precision; (4) the aggregation of textbooks in the same domain increases the coverage of the model significantly.

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, we have developed lists for textbooks written in English, German, French, Spanish, and Dutch. 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 (Alpizar-Chacon & Sosnovsky, 2019). Another application that uses our approach as foundation is Intextbooks: a system capable of transforming PDF textbooks into intelligent educational resources (Alpizar-Chacon et al., 2020).

We plan to further extend the approach and improve extraction of models from a wider range of textbooks and domains. In particular, it is interesting to explore how to maintain high accuracy of the composite model when more and more textbooks are integrated. Some mechanism to enforce consensus among individual glossaries need to be implemented. 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 behaviour 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. Finally, on a grander scale, our 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.

The developed rule-based model extraction system is available as a GitHub project at https://github.com/intextbooks/ITCore and as a web service at https://intextbooks.science.uu.nl/.

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

Isaac Alpizar-Chacon http://orcid.org/0000-0002-6931-9787
Sergey Sosnovsky http://orcid.org/0000-0001-8023-1770

References

The chicago manual of style. 2017. The University of Chicago Press.

Agirre, E., & Rigau, G. (1996). Word sense disambiguation using conceptual density. In J. Tsujii (Ed.), *Proceedings of the 16th Conference on Computational Linguistics* (Vol. 1, p. 8). Association for Computational Linguistics.

Alpizar-Chacon, I., & Sosnovsky, S. (2019). Interlingua: Linking textbooks across different languages. In S. Sosnovsky, P. Brusilovsky, R. Baraniuk, R. Agrawal, & A. Lan (Eds.), *Proceedings of the First Workshop on Intelligent Textbooks* (Vol. 2384, pp. 104–117). CEUR-WS.

Alpizar-Chacon, I., van der Hart, M., Wiersma, Z. S., Theunissen, L., & Sosnovsky, S. (2020). Transformation of PDF textbooks into intelligent educational resources. In S. Sosnovsky, P. Brusilovsky, R. Baraniuk, & A. Lan (Eds.), *Proceedings of the Second Workshop on Intelligent Textbooks* (Vol. 2674, pp. 4–16). CEUR-WS.

Ament, K. (2001). *Indexing: A nuts-and-bolts guide for technical writers*. William Andrew.

Baker, J. B., Sexton, A. P., & Sorge, V. (2009). A linear grammar approach to mathematical formula recognition from PDF. In J. Carette, L. Dixon, C. Sacerdoti Coen, & S. M. Watt (Eds.), *Intelligent computer mathematics* (pp. 201–216). Springer.

Bast, H., & Korzen, C. (2017). A benchmark and evaluation for text extraction from PDF. In *Proceedings of the 17th ACM/IEEE Joint Conference on Digital Libraries* (pp. 99–108). IEEE Press.

Bayomi, M., & Lawless, S. (2018). C-HTS: A concept-based hierarchical text segmentation approach. In N. Calzolari (Conference chair), K. Choukri, C. Cieri, T. Declerck, S. Goggi, K. Hasida, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, S. Piperidis, & T. Tokunaga (Eds.), *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*. European Language Resources Association.

Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., & Hellmann, S. (2009). DBpedia: A crystallization point for the Web of Data. *Journal of Web Semantics*, *7*(3), 154–165. https://doi.org/10.1016/j.websem.2009.07.002

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, *3*(Jan), 993–1022.

Bouarroudj, W., & Boufaida, Z. (2018). A candidate generation algorithm for named entities disambiguation using DBpedia. In Á. Rocha, H. Adeli, L. Paulo Reis, & S. Costanzo (Eds.), *Trends and advances in information systems and technologies* (pp. 712–721). Springer.

Chambliss, M. J. (2002). The characteristics of well-designed science textbooks. In J. Otero, J. A. León, & A. C. Graesser (Eds.), *The psychology of science text comprehension* (pp. 51–72).

Chang, A. X., Spitkovsky, V. I., Manning, C. D., & Agirre, E. (2016). A comparison of named-entity disambiguation and word sense disambiguation. In N. Calzolari, K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, & S. Piperidis (Eds.), *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)* (pp. 860–867). European Language Resources Association.

Chao, H., & Fan, J. (2004). Layout and content extraction for PDF documents. In S. Marinai & A. R. Dengel (Eds.), *Document analysis systems VI* (Vol. 3163, pp. 213–224). Springer.

Councill, I. G., & Giles, C. L. (2008). ParsCit: An open-source CRF reference string parsing package. In N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, & D. Tapias (Eds.), *International language resources and evaluation*. European Language Resources Association.

Dalgaard, P. (2011). *Introductory statistics with R*. Lightning Source UK Ltd.

Déjean, H., & Meunier, J. L. (2009). On tables of contents and how to recognize them. *International Journal of Document Analysis and Recognition (IJDAR)*, *12*(1), 1–20. https://doi.org/10.1007/s10032-009-0078-8

Dekking, F. M., Kraaikamp, C., Lopuhaä, H. P., & Meester, L. E. (2005). *A modern introduction to probability and statistics: Understanding why and how*. Springer.

Demartini, G., Difallah, D. E., & Cudré-Mauroux, P. (2013, October). Large-scale linked data integration using probabilistic reasoning and crowdsourcing. *The VLDB Journal*, *22*(5), 665–687. https://doi.org/10.1007/s00778-013-0324-z

Devore, J. L., & Berk, K. N. (2012). *Modern mathematical statistics with applications*. Springer.

Doucet, A., Kazai, G., Colutto, S., & Mühlberger, G. (2013). ICDAR 2013 competition on book structure extraction. In L. O'Conner (Ed.), *2013 12th International Conference on Document Analysis and Recognition* (pp. 1438–1443). IEEE Computer Society.

Doucet, A., Kazai, G., Dresevic, B., Uzelac, A., Radakovic, B., & Todic, N. (2011). Setting up a competition framework for the evaluation of structure extraction from OCR-ed books. *International Journal on Document Analysis and Recognition (IJDAR)*, *14*(1), 45–52. https://doi.org/10.1007/s10032-010-0127-3

Dwarakanath, A., Ramnani, R. R., & Sengupta, S. (2013). Automatic extraction of glossary terms from natural language requirements. In *2013 21st IEEE International Requirements Engineering Conference (RE)*, Conference Publishing Consulting, D-94034 Passau, Germany (pp. 314–319). IEEE Computer Society.

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

Faber, M. H. (2012). *Statisticcs and probability theory: In pursuit of engineering decision support*. Springer Verlag.

Fang, J., Gao, L., Bai, K., Qiu, R., Tao, X., & Tang, Z. (2011). A table detection method for multipage pdf documents via visual separators and tabular structures. In P. Kellenberger (Ed.), *2011 International Conference on Document Analysis and Recognition* (pp. 779–783). IEEE.

Färber, M., Ell, B., Menne, C., & Rettinger, A. (2015). A comparative survey of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO. *Semantic Web Journal*, *1*(1), 1–5.

Ferragina, P., & Scaiella, U. (2012). Fast and accurate annotation of short texts with wikipedia pages. *IEEE Software*, *29*(1), 70–75. https://doi.org/10.1109/MS.2011.122

Finkelstein, M. O. (2009). *Basic concepts of probability and statistics in the law*. Springer.

Gao, L., Tang, Z., Lin, X., Liu, Y., Qiu, R., & Wang, Y. (2011). Structure extraction from PDF-based book documents. In *Proceedings of the 11th Annual International ACM/IEEE Joint Conference on Digital Libraries* (pp. 11–20). Association for Computing Machinery.

Gao, L., Tang, Z., Lin, X., & Qiu, R. (2008). Comprehensive global typography extraction system for electronic book documents. In K. Kise & H. Sake (Eds.), *2008 the Eighth IAPR International Workshop on Document Analysis Systems* (pp. 615–621). IEEE Computer Society.

Gao, L., Tang, Z., Lin, X., Tao, X., & Chu, Y. (2009). Analysis of book documents' table of content based on clustering. In B. Werner (Ed.), *Proceedings of the International Conference on Document Analysis and Recognition* (pp. 911–915). IEEE Computer Society.

Giannini, S., Colucci, S., Donini, F. M., & Di Sciascio, E. (2015). A logic-based approach to named-entity disambiguation in the web of data. In M. Gavanelli, E. Lamma, & F. Riguzzi (Eds.), *2015 Advances in Artificial Intelligence* (pp. 367–380). Springer.

Guerra, J., Sosnovsky, S., & Brusilovsky, P. (2013). When one textbook is not enough: Linking multiple textbooks using probabilistic topic models. In D. Hernández-Leo, T. Ley, R. Klamma, A. Harrer (Eds.), *European Conference on Technology Enhanced Learning* (pp. 125–138). Springer.

Hahm, Y., Park, J., Lim, K., Kim, Y., Hwang, D., & Choi, K. S. (2014). Named entity corpus construction using wikipedia and DBpedia ontology. In N. Calzolari, K. Choukri, T. Declerck, H. Loftsson, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, & S. Piperidis (Eds.), *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)* (pp. 2565–2569). European Language Resources Association.

Hall, K. M., Sabey, B. L., & McClellan, M. (2005). Expository text comprehension: Helping primary-grade teachers use expository texts to full advantage. *Reading Psychology*, *26*(3), 211–234. https://doi.org/10.1080/02702710590962550

Han, X., & Sun, L. (2011). A generative entity-mention model for linking entities with knowledge base. In D. Lin, Y. Matsumoto, & R. Mihalcea (Eds.), *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (Vol. 1, pp. 945–954). Association for Computational Linguistics.

Hassan, T. (2009). Object-level document analysis of PDF files. In *Proceedings of the 9th ACM Symposium on Document Engineering*. ACM.

Hoffart, J., Seufert, S., Nguyen, D. B., Theobald, M., & Weikum, G. (2012). KORE: Keyphrase overlap relatedness for entity disambiguation. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management* (pp. 545–554). ACM.

Hollingsworth, B., Lewin, I., & Tidhar, D. (2005). Retrieving hierarchical text structure from typeset scientific articles: A prerequisite for e-science text mining. In S. Cox & D. W. Walker (Eds.), *Proceedings of the 4th UK E-science All Hands Meeting* (pp. 267–273). EPSRC.

Hulpus, I., Prangnawarat, N., & Hayes, C. (2015). Path-based semantic relatedness on linked data and its use to word and entity disambiguation. In M. Arenas, O. Corcho, E. Simperl, M. Strohmaier, M. d'Aquin, K. Srinivas, P. Groth, M. Dumontier, J. Heflin, K.

Thirunarayan, K. Thirunarayan, & S. Staab (Eds.), *Proceedings of the 14th International Semantic Web Conference (ISWC 2015)* (Vol. 9366, pp. 442–457). Springer.

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

Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, *20*(4), 422–446. https://doi.org/10.1145/582415.582418

Kaltenbach, H. M. (2012). *A concise guide to statistics*. Springer.

Kern, R., Jack, K., & Hristakeva, M. (2012, July). TeamBeam - Meta-data extraction from scientific literature. *D-Lib Magazine*, *18*(7/8), 1–1. https://doi.org/10.1045/dlib.magazine

Kern, R., & Klampfl, S. (2013). Extraction of references using layout and formatting information from scientific articles. *D-Lib Magazine*, *19*(9/10), 1–1. https://doi.org/10.1045/dlib.magazine.

Kobilarov, G., Scott, T., Raimond, Y., Oliver, S., Sizemore, C., Smethurst, M., Bizer, C., & Lee, R. (2009). Media meets semantic web: How the BBC uses DBpedia and linked data to make connections. In L. Aroyo, P. Traverso, F. Ciravegna, P. Cimiano, T. Heath, E. Hyvönen, R. Mizoguchi, E. Oren, M. Sabou, & E. Simperl (Eds.), *Proceedings of the 6th European Semantic Web Conference on the Semantic Web: Research and Applications* (pp. 723–737). Springer.

Larrañaga, M., Conde, A., Calvo, I., Elorriaga, J. A., & Arruarte, A. (2014). Automatic generation of the domain module from electronic textbooks: Method and validation. *IEEE Transactions on Knowledge and Data Engineering*, *26*(1), 69–82. https://doi.org/10.1109/TKDE.2013.36

Larrañaga, M., Rueda, U., Elorriaga, J. A., & Arruarte, A. (2004). Acquisition of the domain structure from document indexes using heuristic reasoning. In J. C. Lester, R. M. Vicari, & F. Paraguaçu (Eds.), *Intelligent tutoring systems* (pp. 175–186). Springer Berlin Heidelberg.

Lin, X., Gao, L., Tang, Z., Lin, X., & Hu, X. (2011). Mathematical formula identification in PDF documents. In P. Kellenberger (Ed.), *Proceedings of the International Conference on Document Analysis and Recognition* (pp. 1419–1423). IEEE.

Liu, B. (2007). *Uncertainty theory*. Springer.

Lopes, L., Vieira, R., Finatto, M. J., & Martins, D. (2010, November). Extracting compound terms from domain corpora. *Journal of the Brazilian Computer Society*, *16*(4), 247–259. https://doi.org/10.1007/s13173-010-0020-4

Madsen, B. (2011). *Statistics for non-statisticians*. Springer.

Manning, C. D., Raghavan, P., & Schütze, H. (2009). *An introduction to information retrieval*. Cambridge University Press.

Marie, N., Gandon, F., Ribière, M., & Rodio, F. (2013). Discovery hub: On-the-fly linked data exploratory search. In M. Sabou, E. Blomqvist, T. Di Noia, H. Sack, & T. Pellegrini (Eds.), *Proceedings of the 9th International Conference on Semantic Systems* (pp. 17–24). ACM.

Marinai, S., Marino, E., & Soda, G. (2010). *Table of contents recognition for converting PDF documents in e-book formats. Proceedings of the 10th ACM Symposium on Document Engineering* (pp. 73–76). Association for Computing Machinery. https://doi.org/10.1145/1860559.1860576

Medelyan, O., Witten, I. H., & Milne, D. (2008). Topic indexing with Wikipedia. In R. Bunescu, E. Gabrilovich, & R. Mihalcea (Eds.), *Proceedings of the AAAI Wikiai Workshop* (Vol. 1, pp. 19–24). Association for the Advancement of Artificial Intelligence.

Mendes, P. N., Jakob, M., García-Silva, A., & Bizer, C. (2011). DBpedia spotlight: Shedding light on the web of documents. In C. Ghidini, A.-C. Ngonga Ngomo, S. Lindstaedt, & T.

Pellegrini (Eds.), *Proceedings of the 7th International Conference on Semantic Systems* (pp. 1–8). ACM.

Milne, D., & Witten, I. H. (2008). Learning to link with wikipedia. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management* (pp. 509–518). ACM.

Mirizzi, R., Ragone, A., Di Noia, T., & Di Sciascio, E. (2010a). Ranking the linked data: The case of DBpedia. In B. Benatallah, F. Casati, G. Kappel, & G. Rossi (Eds.), *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)* (Vol. 6189 LNCS, pp. 337–354). Springer.

Mirizzi, R., Ragone, A., Di Noia, T., & Di Sciascio, E. (2010b). Semantic wonder cloud: Exploratory search in DBpedia. In F. Daniel & F. Michele Facca (Eds.), *International Conference on Web Engineering* (pp. 138–149). Springer.

Moro, A., Raganato, A., & Navigli, R. (2014). Entity linking meets word sense disambiguation: A unified approach. *Transactions of the Association for Computational Linguistics*, *2* (22), 231–244. https://doi.org/10.1162/tacl_a_00179

Navigli, R., & Ponzetto, S. P. (2012). BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, *193* (December), 217–250. https://doi.org/10.1016/j.artint.2012.07.001

Oro, E., & Ruffolo, M. (2009). PDF-TREX: An approach for recognizing and extracting tables from PDF documents. In B. Werner (Ed.), *Proceedings of the International Conference on Document Analysis and Recognition* (pp. 906–910). IEEE Computer Society.

Ramakrishnan, C., Patnia, A., Hovy, E., & Burns, G. A. (2012). Layout-aware text extraction from full-text PDF of scientific articles. *Source Code for Biology and Medicine*, *7*(1), Article number: 7. https://doi.org/10.1186/1751-0473-7-7

Ramanathan, C., Jayabal, Y., & Sheth, M. J. (2012). Challenges in generating bookmarks from TOC entries in e-books. In *Proceedings of the 2012 ACM Symposium on Document Engineering* (p. 37). ACM.

Ramnarayan, P., Tomlinson, A., Rao, A., Coren, M., Winrow, A., & Britto, J. (2003). ISABEL: A web-based differential diagnostic aid for paediatrics: Results from an initial performance evaluation. *Archives of Disease in Childhood*, *88*(5), 408–413. https://doi.org/10.1136/adc.88.5.408

Rodríguez Rocha, O., Faron Zucker, C., & Giboin, A. (2018, June). Extraction of relevant resources and questions from DBpedia to automatically generate quizzes on specific domains. In R. Nkambou, R. Azevedo, & J. Vassileva (Eds.), *Intelligent Tutoring Systems* (pp. 380–385). Springer.

Ruiz Fabo, P., Bermúdez Sabel, H., Martínez Cantón, C. I., González-Blanco García, E., & Navarro Colorado, B. (2018). The diachronic Spanish sonnet corpus (DISCO): TEI and linked open data encoding, data distribution and metrical findings. In J. Girón Palau & I. Galina Russell (Eds.), *Digital humanities 2018, book of abstracts*. Red de Humanidades Digitales A. C.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, *24*(5), 513–523. https://doi.org/10.1016/0306-4573(88)90021-0

Salton, G., Wong, A., & Yang, C. S. (1975, November). A vector space model for automatic indexing. *Communications of the ACM*, *18*(11), 613–620. https://doi.org/10.1145/361219.361220

Shanmugam, R., & Chattamvelli, R. (2015). *Statistics for scientists and engineers*. Wiley.

Shao, M., & Futrelle, R. P. (2005). Recognition and classification of figures in PDF documents. In W. Liu & J. Lladós (Eds.), *International Workshop on Graphics Recognition* (pp. 231–242). Springer.

Slabbekoorn, K., Hollink, L., & Houben, G. J. (2012, November). Domain-aware ontology matching. In P. Cudré-Mauroux, J. Heflin, E. Sirin, T. Tudorache, J. Euzenat, M. Hauswirth, J. Xavier Parreira, J. Hendler, G. Schreiber, A. Bernstein, & E. Blomqvist (Eds.), *The Semantic Web-ISWC 2012* (pp. 542–558). Springer.

Sosnovsky, S., Hsiao, I. H., & Brusilovsky, P. (2012). Adaptation 'in the wild': Ontology-based personalization of open-corpus learning material. In A. Ravenscroft, S. Lindstaedt, C. Delgado Kloos, & D. Hernández-Leo (Eds.), *European Conference on Technology Enhanced Learning* (pp. 425–431). Springer.

Stahn, L. L., Hennicke, S., & De Luca, E. W. (2016). Using TEI for textbook research. In E. Hinrichs, M. Hinrichs, & T. Trippel (Eds.), *Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities* (pp. 181–186). The COLING 2016 Organizing Committee.

Tittel, S., Bermúdez-Sabel, H., & Chiarcos, C. (2018). Using RDFa to link text and dictionary data for medieval French. In N. Calzolari, K. Choukri, C. Cieri, T. Declerck, S. Goggi, K. Hasida, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, S. Piperidis, & T. Tokunaga (Eds.), *Proceedings of the Eleventh International Conference on Language Resources and Evaluation* (pp. 7–12). European Language Resources Association.

Tkaczyk, D., Szostek, P., Fedoryszak, M., Dendek, P. J., & Bolikowski, Ł. (2015). CERMINE: Automatic extraction of structured metadata from scientific literature. *International Journal on Document Analysis and Recognition*, *18*(4), 317–335. https://doi.org/10.1007/s10032-015-0249-8

Tuarob, S., Bhatia, S., Mitra, P., & Giles, C. L. (2013, August). Automatic detection of pseudocodes in scholarly documents using machine learning. In L. O'Conner (Ed.), *2013 12th International Conference on Document Analysis and Recognition* (pp. 738–742). IEEE.

Tuarob, S., Bhatia, S., Mitra, P., & Giles, C. L. (2016, March). AlgorithmSeer: A system for extracting and searching for algorithms in scholarly big data. *IEEE Transactions on Big Data*, *2*(1), 3–17. https://doi.org/10.1109/TBDATA.2016.2546302

Tuarob, S., Mitra, P., & Giles, C. L. (2015). A hybrid approach to discover semantic hierarchical sections in scholarly documents. In *2015 13th International Conference on Document Analysis and Recognition* (pp. 1081–1085). IEEE Computer Society.

Ubøe, J. (2017). *Introductory statistics for business and economics: Theory, exercises and solutions*. Springer International Publishing AG.

Usbeck, R., Ngomo, A. C. N., Röder, M., Gerber, D., Coelho, S. A., Auer, S., & Both, A. (2014). AGDISTIS - Graph-based disambiguation of named entities using linked data. In P. Mika, T. Tudorache, A. Bernstein, C. Welty, C. Knoblock, D. Vrandečić, P. Groth, N. Noy, K. Janowicz, & C. Goble (Eds.), *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)* (Vol. 8796, pp. 457–471). Springer.

Vrandečić, D., & Krötzsch, M. (2014, September). Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, *57*(10), 78–85. https://doi.org/10.1145/2629489

Walpole, R. E., Myers, R. H., Myers, S. L., & Ye, K. (2012). *Probability & statistics for engineers & scientists*. Prentice Hall.

Wang, S., Liang, C., Wu, Z., Williams, K., Pursel, B. K., Brautigam, B., Saul, S., Williams, H., Bowen, K., & Giles, C. L. (2015). Concept hierarchy extraction from textbooks. In *Proceedings of the 2015 ACM Symposium on Document Engineering* (pp. 147–156). ACM.

Whitington, J. (2011). PDF explained. In (chap. I Introduction). O'Reilly Media.

Wu, Z., Das, S., Li, Z., Mitra, P., & Giles, C. L. (2013). Searching online book documents and analyzing book citations. In *Proceedings of the 2013 ACM Symposium on Document Engineering* (p. 81). ACM.

Wu, Z., Mitra, P., & Giles, C. L. (2013, August). Table of contents recognition and extraction for heterogeneous book documents. In L. O'Conner (Ed.), *2013 12th International Conference on Document Analysis and Recognition* (pp. 1205–1209). IEEE.

Wu, J., Williams, K. M., Chen, H. H., Khabsa, M., Caragea, C., Tuarob, S., Ororbia, A. G., Jordan, D., Mitra, P., & Giles, C. L. (2015). CiteSeerX: AI in a digital library search engine. *AI Magazine*, *36*(3), 35–48. https://doi.org/10.1609/aimag.v36i3.2601