From Books to Knowledge Graphs

Natallia Kokash[*1], **Matteo Romanello**[2], **Ernest Suyver**[3], **Giovanni Colavizza**[1]

[1]University of Amsterdam, Amsterdam, The Netherlands
[2]University of Lausanne, Lausanne, Switzerland
[3]Brill, Leiden, The Netherlands

Abstract

The digital transformation of the scientific publishing industry has led to dramatic improvements in content discoverability and information analytics. Unfortunately, these improvements have not been uniform across research areas. The scientific literature in the arts, humanities and social sciences (AHSS) still lags behind, in part due to the scale of analog backlogs, the persisting importance of national languages, and a publisher ecosystem made of many, small or medium enterprises. We propose a bottom-up approach to support publishers in creating and maintaining their own publication knowledge graphs in the open domain. We do so by releasing a pipeline able to extract structured information from the bibliographies and indexes of AHSS publications, disambiguate, normalize and export it as linked data. We test the proposed pipeline on Brill's Classics collection, and release an implementation in open source for further use and improvement.

I INTRODUCTION

The scientific publishing industry has transitioned to information analytics. Researchers immensely benefit from the indexed content and provision of advanced search engines, primarily based on scientific publication metadata, citations and, increasingly, contents. Notable examples include Google Scholar, Semantic Scholar, Dimensions, and more [Martín-Martín et al., 2021, Visser et al., 2021]. Advances in the adoption of the open science agenda have resulted in the increased availability of open and structured scientific literature data, for example via PubMed [White, 2020], OpenAlex[1] and OpenCitations [Peroni and Shotton, 2020]. Unfortunately, this digital transformation is not occurring uniformly in all research areas. A divide persists for the Arts, Humanities and, to a lesser degree, the Social Sciences (AHSS): the indexation of this literature is still lagging, in particular with respect to historical backlogs and publications in languages other than English [Colavizza and Romanello, 2019], both essential parts of it [Kulczycki et al., 2018, 2020]. Recently, proposals have been made to better support LAM organisations (Libraries, Archives, Museums) in the indexation of the literature they collect [Colavizza et al., 2018, 2021]. Another obstacle to a better indexation of AHSS literature is that the publishing ecosystem supporting these disciplines consists of many specialized organizations that cannot, individually, transform established practices and develop the large-scale services required for systematic indexation. We address this latter challenge here.

We propose to start bridging the gap by supporting small and medium AHSS publishers to create and maintain their own _knowledge graphs_ (KG): the technology underpinning modern

---

[1]https://openalex.org.

scientific search engines [Luan et al., 2018, Jaradeh et al., 2019]. We do so by releasing an end-to-end information extraction pipeline which takes unstructured publications as input and outputs structured, relational information. The proposed pipeline extracts the relevant information, performs entity linking to equip them with identifiers, and structures information following the SPAR and OpenCitations data models [Peroni and Shotton, 2018, Daquino et al., 2020]. The resulting linked data is made ready for ingestion into OpenCitations. AHSS literature, composed to a significant degree of books, already contains rich and well-structured information – such as back matter contents including references and indexes – which we leverage, automatically mine and interlink [Romanello, 2019]. In openly releasing this pipeline, our aim is to foster its adoption and future extension by AHSS publishers, and contribute to a better indexation of AHSS literature in scientific search engines.

Our work has been conducted in collaboration with Brill, a leading Humanities publisher. The Brill Classics catalogue has been used as a case study for the development of the proposed pipeline. The rest of this paper is structured as follows: Section II presents the challenge of extracting structured information from AHSS publications, Section III presents our proposed pipeline, Section IV discusses the resulting Knowledge Graph, while Sections V and VI conclude with an evaluation of the pipeline and the discussion of related literature.

II CONTENT REPRESENTATION

In this section we describe the structure and format of AHSS publications, and the Brill's corpus we used as a case study.

2.1 The structure of AHSS publications

An AHSS book (monograph or edited volume) typically consists of *front matter*, *body*, and *back matter*. The *front matter* usually contains a title page, copyright page, table of contents, and one or more personal preface or introductions. The *body* or main text typically consist of a number of chapters or book sections, among which the first (introduction or prologue) and the last (conclusions or epilogue) stand out as they open the narrative and sum up the core ideas. The *back matter* may contain appendices, indexes and a bibliography. These supplementary sections are meant to inform the reader about certain aspects of the content, and the choices largely depend on each particular book's design.

The front and back matter are the parts of a book we are most interested in the context of building a KG that links a given publication with related works and subjects. The front matter provides metadata necessary to identify a publication: author, title, publication year, and (for modern works) persistent identifiers like ISBN (International Standard Book Number) or DOI (Digital Object Identifier).

The analysis of the back matter of published literature in the AHSS domain was the most challenging aspect we dealt with due to the variety of its content, formats, conventions, and organization rules. Two types of the back matter sections particularly relevant to us are:
- *Bibliography*: a list of references, usually to secondary literature. Primary sources are, instead, often listed in an index.
- *Index*: a curated list of relevant items and their mentions in the book.

If the book is a *monograph*, i.e., written by one author, the bibliography tends to be at the back of the book. If the book is an *edited volume*, i.e., it contains chapters written by different authors, the bibliography tends to be at the end of each chapter. Indexes are usually found at the

Bibliography

Accame, S. (1963). "L'invocazione alle Muse e la Verità in Omero ed in Esiodo", RFIC 91: 257–81, 385–415.

Accorinti, D. (2004). *Nonno di Panopoli. Le Dionisiache 4, Canti XI.–XLVIII* (Milan).

Accorinti, D. and Chuvin P. (eds.), *Des Géants à Dionysos. Mélanges de mythologie et de poésie grecques offerts à Francis Vian* (Alessandria).

Acosta-Hughes, B., Kosmetatou, E., and Baumbach, M. (eds.) (2004). *Labored in Papyrus Leaves: Perspectives on an Epigram Collection Attributed to Posidippus (P. Mil. Vogl. VIII 309)* (Washington, D.C.).

Adriani, A. (1944). "L' 'Afrodite al Bagno' di Rodi e l'Afrodite di Doedalsas", ASAE XLIV: 37–70.

——— (1951). "Contributo allo Studio dell' Afrodite di Doedalsas", BSRAA 39: 144–82.

Figure 1: Bibliography references (DOI: 10.1163/9789004289598_008)

back of the book, but not all books have them. Index types may vary: sometimes there is only one, sometimes, there are several. A usual pattern is an index of names (*index nominum*), an index of places (*index locorum*)[2] and an index of things, i.e., subjects (*index rerum*). Depending on the subject, there can be more specialized indexes.

Figure 1 shows a fragment of back matter with bibliography references. Figure 2 shows examples of index sections: an index of general terms (Figure 2(a)) and an index of places (Figure 2(b)).

---

[2]Note that an index of places is not about place names (geolocations), but about passages in ancients texts (the so-called *loci*).

General Index

Acamas	174 n. 168, 241
Acastus	232
Acathistus hymn	20
Achilles	74 n. 153, 75–76, 77 n. 173, 90–2, 111 n. 282, 121, 129–30, 137, 153, 170 n. 151, 194, 199, 212 n. 337, 342, 265, 270
Acontius	95, 242
Acrotime	53
Actaeon	248
Admetus	210 n. 326, 214, 216, 218
Adonis	105 n. 259
Aeacus	194
Aeetes	57, 76 n. 172

Index Locorum

Ach. Tat.			
		9.154.1	237
1.1–2	58 n. 86	11.372	112 n. 287
1.4.4	181		
1.4.5	184	**Alc.**	
1.9.1	182	fr. 42	255
1.9.4–5	180 n. 211, 184	fr. 283	212 n. 389
2.3	234 n. 53	fr. 691	237 n. 67
2.7.6	195 n. 265		
2.13.1–2	182	**[Alex. Aphr.]** *Probl.*	
5.25.7–8	162 n. 118	2.42	179 n. 201
6.1.3	162 n. 118		
8.6.4	71 n. 138	**Ammian.**	
		22.6.1	114

(a) DOI: 10.1163/9789004289598_010   (b) DOI: 10.1163/9789004289598_009

Figure 2: Index references

2.1.1 Citations and bibliographic references

References in bibliographies must be distinguished from citations in the text. A citation is an often-abbreviated pointer to a location in a publication. The book is usually identified by a combination of author name and publication year. The reference does not give the location, but instead gives the complete bibliographical data for that publication. For example: on p. 13, footnote 63 of our example book Cadau [2015], we find this citation:

> Miguélez-Cavero 2008, 87.

This corresponds to the following reference in the bibliography on page 292:

> ——— (2008). Poems in Context: Greek Poetry in the Egyptian Thebaid 200–600 AD (Berlin).

Note that the author name is replaced by three dashes. This is because the author name is already mentioned in a previous reference. This is a matter of citation style: many such styles exist. Most Brill book series follow the Chicago Author-Date style. [3]

2.1.2 Indexes

The first two items in the *general index* of the same example read:

> Acamas 174 n. 168, 241
> Acastus 232

This is a two-column table: on the left are the index terms, on the right are the page numbers (locations in the book). Multiple locations are separated by a comma, the left and right parts are separated by a tab. The presence of 'n.' in the location part indicates that the index entry refers to a footnote. To save space, such two-column definitions are often arranged into several columns on every page.

The *index locorum* tells us which passage (*locus*), from which ancient work occurs on which page of the work in question. For example, the following index entry

> Alc. fr. 42 255 fr. 283 212 n. 389 fr. 691 237 n. 67

refers to the name of the author, the poet *Alcaeus*, in abbreviated form, on the left, and a set of his fragment occurrences, on the right. No specific work of his is mentioned in the book, only fragments (hence, abbreviation *fr.*). In the cases where we do have the works, they are commonly mentioned in an abbreviated form too:

> Aesch. Ag. 681–98 82

2.2 Structured annotations

In order to extract useful data from the publication sources, we first need to locate its relevant parts. If the work were given in a single file (e.g., PDF), the only option would be to analyze the text and the layout of the file to distinguish front matter, body, and back matter. However, the essential metadata such as author's name, title, tables of contents, etc., is part of the publication and dissemination processes and is normally kept by publishers in a structured form.

The *Journal Article Tag Suite* JATS is a standard XML vocabulary designed to model current journal articles. It provides a named collection of XML elements and attributes that can be used to mark the structure and semantics of a single journal article. Originally, JATS was used for

---

[3] https://www.chicagomanualofstyle.org.

STEM (Scientific, Technical, Engineering and Medical) articles, but now publishers in AHSS also widely adopt this markup language. Tools have been developed to export JATS out of many other formats, including Microsoft Word and LaTeX, and to generate quality PDF, HTML, and various eBook formats out of it.

Book collections are often annotated using a JATS extension called *Book Interchange Tag Set* BITS. The intent of BITS is to provide a common format for publishers and archives to exchange book content. The tag set describes both the metadata and the narrative content of a book and its components, as well as collection-level metadata for book sets and series. The BITS annotation may include the following (optional) components:

- *Collection metadata*: the `<collection-meta>` element describes book sets or series to which this book or book part belongs.
- *Book metadata*: the `<book-meta>` element includes nested tags defining, for example, the title (`<book-title-group>`), the contributors (`<contrib-group>`), the date of publication (`<pub-date>`), the publisher (`<publisher>`), etc.
- *Front matter*: the `<front-matter>` element provides the textual front material for a book, such as a dedication, foreword, or preface.
- *Body of the book*: the `<book-body>` element defines the structure of its main textual and graphic content. The body of a book contains book parts `<book-part>` (which may be called parts, sections, chapters, modules, lessons, or whatever divisions a publisher uses). Book parts are recursive and may contain other book parts.
- *Back matter*: the `<book-back>` element contains the ancillary information such as lists of references (`<ref-list>`) and indexes (`<index-group>`).

Although the JATS/BITS format provides tags for bibliography and index representation, in our case study, these data is not available in this structured form and must be retrieved from PDF files.

2.3 Brill's case study

The dataset used for the creation of our pipeline and a KG comprised of books in the field of Classics. This field was selected because of the Brill's domain expertise and experience with information extraction of this type of content. We chose books over other publication types because they are characteristic of the AHSS and because, unlike journal articles, they have extensive back-of-the-book indexes that can be used for content discovery.

The corpus consists of 1816 books with an average of 369 pages. The books were produced in the period 2006-2021. The books included 965 edited volumes (collections of chapters by different authors), and 851 monographs (books on one subject written by a single author).

At Brill, books are archived in the following manner. A folder is created using the ISBN for the online version ('e-ISBN'). It contains content and metadata and is archived in a compressed form. The content is usually in the form of a single PDF for the entire book, as well as PDFs for the individual book sections, and metadata are in BITS XML, for example:

```
9789004339460_BITS.zip
├── 9789004339460_webready_content_s001.pdf
├── ...
├── 9789004339460_webready_content_s018.pdf
├── 9789004339460_webready_content_text.pdf
└── 9789004339460_webready_content_text.xml
```

A large obstacle, characteristic for small-to-medium-sized publishers, is the lack of consistency in the typography and the content due to the following factors:

- *Content variability*. Books have different subjects; authors belong to different fields; countries, even institutions, have their own conventions. Brill books usually belong to series, and each series has its own typographical, orthographic, and other conventions. Within the series, volumes may differ. Moreover, the entire workflow (writing, peer review, copy editing, indexing) is manual and lacks the use of standardized tools.
- *Choice of typesetters*. Books are produced by different typesetters, e.g., using InDesign, PageMaker, or LaTeX. Until 2018, there was no Brill Typographical Style (BTS) [van Waarden, 2020]. Even now, BTS, despite its name, is not a uniform typographical style applied to all Brill content, but a limited set of instructions. The execution of BTS is left to the typesetters. BTS allows the idiosyncrasies of the individual series to continue, e.g., BTS does not prescribe a citation style.
- *Publishing process evolution*. In 2006, the number of books produced was less then a third of what is now [van der Veen, 2008]. A large number of small typesetters were used, and author PDFs were common. As time went by and the volume of content increased, more uniformity was imposed on the workflow. In particular, tools were introduced for submission, ERP (Enterprise Resource Planning), workflow, and typesetting, including the Brill typeface BrillTypeface.

Manuscripts are mostly submitted in the form of MS Word documents. Authors are responsible for the creation of the citations and references. In most cases, they are also responsible for the creation of indexes, and rarely use any specialized tools for this. Ultimately, all typographical information is stripped by the typesetter and rebuilt using their own systems. The typeset manuscript is returned to the author for proofreading. A number of proofs (usually two or three) is submitted and returned in what is still a manual process. The final result is a carefully proofread and typeset PDF, published in print and online. The same PDF is archived and distributed to discovery services, together with its metadata.

The catalogue we used reflects the challenges Brill faces on the way of transitioning to information analytics, and that these challenges are characteristic of other AHSS publications and publishers.

III KNOWLEDGE RETRIEVAL PIPELINE

Keeping in mind the organizational structures and assumptions outlined in the previous section, we start our knowledge graph construction from locating and analyzing relevant publication parts. The main steps of our data processing pipeline are illustrated in Figure 3. The figure shows three sub-processes, each demarcated by start (white circle) and end (grey circle) events, and data exchange between them.

The first process is dedicated to data extraction and database population. At this stage, we access a publisher's catalogue and analyze the structural content of individual publications in it. Since such catalogues can be very large, we assume the need to process data in batches of manageable sizes. The batches can be processed sequentially or in parallel, and the process can be stopped and resumed later in time. It is also useful to be able to redo the extraction of data from selected publications without the need to run the whole pipeline again. Given a publication archive or folder, we extract essential data from its metadata file, front and back matter, and create a structured publication object that then gets serialized and stored in a graph database. The graph-based representation of the publication consists of a node that describes
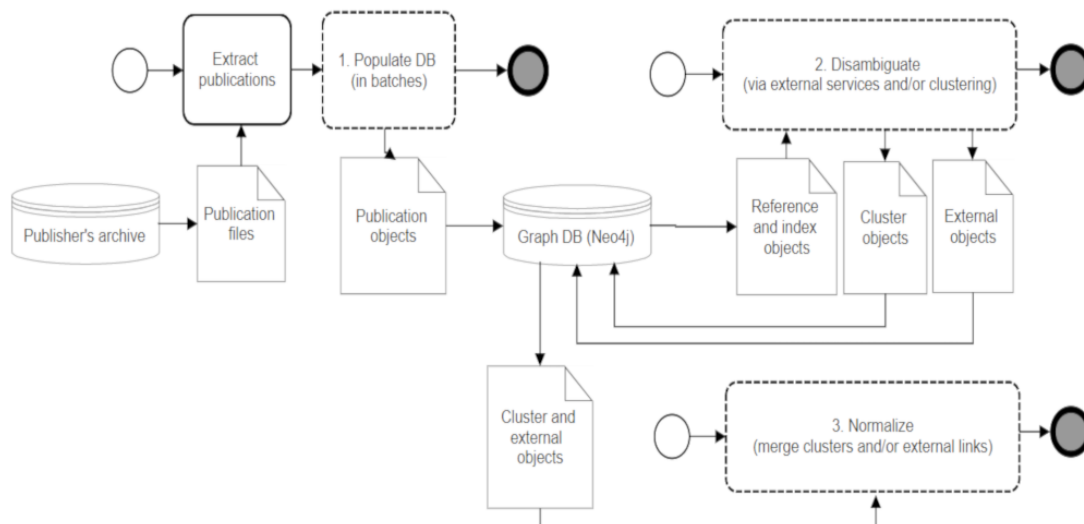
Figure 3: Knowledge graph construction process

its metadata and a set of nodes representing related entities: contributors, identifiers, and, most significantly, bibliographic references and index objects.

The second stage is dedicated to the disambiguation of bibliographic references and index objects created at the step above. This process covers two major procedures:

- firstly, we identify and cluster bibliographic references that correspond to the same published work version and index terms that refer to the same thing (e.g., subject, person).
- Secondly, we match the extracted reference objects with objects representing the same entities in available public authority records.

The output of this stage consists of a set of new graph nodes representing reference clusters or external links related to the bibliographic references or index objects in our database.

Finally, at the third stage, a number of normalization procedures can be performed to merge cluster and external link nodes corresponding to the same reference or index term. Such node duplicates originate from the fact that each batch is processed independently, i.e., we always create necessary graph nodes while in practice it may be possible to link newly added bibliographic reference and index nodes to existing cluster and external link nodes.

3.1 Extracting data from PDF files

For each publication folder, we start from locating the JATS/BITS XML file and extracting metadata identifying the publication: title, contributors (authors and/or editors), publication year, publisher, DOI and ISBN identifiers. Then we search for book parts that contain bibliographic references and index lists. These parts are not marked in any special way in our sample archives, so we rely on keywords *bibliography* and *index* in the book part title to locate the corresponding PDF files.

Additionally, we classify index lists to determine what information they provide. Currently, we recognize the following index types: *verborum* (general), *locorum* (citations), *nominum* (names, ancient and modern), *rerum* (subjects), *geographicus* (geographic locations), *bibliographicus* (manuscripts), *museum* (museums), and *epigraphic* (inscriptions). The classifier relies on a number of hits between the index title and predefined terms commonly used to define such indexes. For example, an index file is likely to be an *index nominum* (index of names) if any
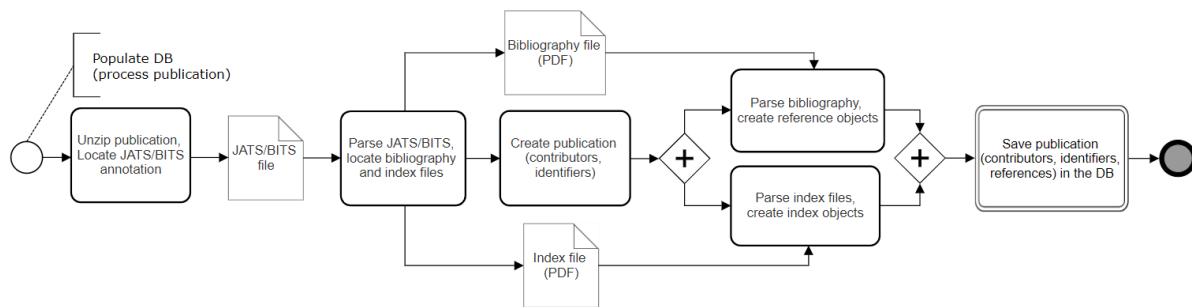
Figure 4: Data extraction pipeline.

of the following words is mentioned in its title: 'nominum', 'nominvm', 'propriorvm', 'name', 'proper', 'person', 'personal', 'people', 'writer', 'poet', 'author', etc. Also, similar terms in other languages are considered: 'auteur', 'eigennamen', 'noms', 'propres', 'personnages', etc. Furthermore, terms like 'ancient', 'antique', 'classical', 'medieval', 'greek', 'egyptian', 'latin', etc. indicate that this is an index of ancient names. The keyword lists are manually created by analyzing titles from a subset of our case study corpus. Some publications may contain highly specialized indices which we cannot properly classify and handle as general. The index types may help parsing index reference text into meaningful parts (i.e., understand that the first word in an entry like *Cicero Att. 1.16.1* is an author name, the next term is their work, and the following numbers are loci) and to help with term disambiguation (e.g., one may want to use a specialized service such as the Classical Works Knowledge Base CWKB for linking this entry to a certain passage by M. Tullius Cicero from 'Epistulae ad Atticum' [4]).

Figure 4 sketches the process of data extraction for the initial KG construction. After locating the bibliography and index files, the key tasks are to split these PDF files into lists of individual bibliographic and index references. Each item from the lists of textual references will then produce structured reference objects serialized as KG nodes.

As was pointed out in Section II, the syntax and the layout of bibliography and index files are rather versatile, there are no standardised ways to represent these data. Even when a popular format such as CMS Author-Date is used across a publication, splitting the bulk data into individual references is still a daunting task: references can be placed in one or several columns, their parts can be separated by commas, different units of indent, alignment, long references can spill into several lines or have nested structure with several sub-levels and relation to above content. However, it can be observed that both bibliographic and index references are *lists*, and, hence, are usually aligned in a uniform way, i.e., each new entry starts from the same horizontal offset as others. This observation also holds for multi-line and multi-level definitions: references that do not fit into one line, resume on the next line, typically, with some extra offset. The same observation is valid for the second column in publications with double-column layout and multi-column index files.

Our main tool for data retrieval is PDFMiner PDFMiner, a text extraction tool for PDF documents which also obtains the exact location of the text as well as other layout information (e.g., font size). PDFs drastically differ from usual text representation formats – it is primarily a graphic representation and contains, essentially, instructions on how to place objects on a display or paper. In most cases, it has no logical structures such as sentences or paragraphs.

---

[4] http://www.perseus.tufts.edu/hopper/text?doc=urn:cts:latinLit:phi0474.phi057.perseus-lat1:1.16.1

However, PDFMiner provides the class `PDFPageInterpreter` that gives access to the page content. Further, we iterate over the `LTTextContainer` elements of the extracted pages, and, given coordinates of the text containing boxes, build a map that relates each unique x-coordinate position with the number of lines starting from it. This is done separately for odd and even pages. Given such a map and assuming that references constitute the majority of the information on pages from a dedicated back matter file, we are able to determine one or more horizontal offset positions where new references start, in the following way:

1. sort the map in descending order according to the number of lines per offset.
2. Mark the left-most position with a large number of lines as the starting position for new references. A map entry with a small offset to the right from this position will indicate the continuation of a long reference or a second level of the index reference (similarly, we can establish the presence of the third level for index definitions).
3. A map entry with a large number of lines and a significant offset (close to the half of the text width) will signify a two-column format. Similarly to the first column, any lines with a small offset from this position represent long reference continuation or index sub-levels.
4. Ignore entries with just a few lines per page: they correspond to 'noise' such as e.g., title, subtitle, page number, footnote.

3.1.1 Parsing bibliographic references

The previous step produces an ordered list of text references per publication, some of which follow the CMS Author-Date format:

- Andrewes, A. 1961, "Philochoros on phratries", Journal of Hellenic Studies, vol. 81, pp. 1–15.
- ——. 1994. "Legal space in classical Athens", *Greece and Rome*, vol. 41, pp. 172–186.
- ...

While others do not:

- Vernant, Jean-Pierre, *Mythe et société en Grèce ancienne* (Paris, 2004).
- Vernant, Jean-Pierre, "One... Two... Three: Eros," in *Before Sexuality: The Construction of Erotic Experience in the Ancient Greek World*, ed. Donald M. Halperin, John J. Winkler, and Froma I. Zeitlin (Princeton, 1999), 465-478.
- ...

However, depending on the purpose of the KG and further processing steps, it may be necessary or desirable to parse such textual definitions into structured objects. In particular, since publications are commonly identified by the author-date pair, it is useful to extract the author name and publication year, and consider the rest of the reference text to be a title. Listing 1 provides a sample implementation of a bibliography reference parser using the Python's Pyparsing library PyParsing: it attempts to retrieve a list of author names (assuming each is defined as a family name followed by one or several, possibly abbreviated, given names), followed by a year (or a year range), and assume the rest of the line constitutes a title. In this grammar, `ppu` is an abbreviation for the `pyparsing_unicode` class that defines various useful sets of Unicode characters. Although not every reference text will be accepted by this grammar, the library makes it easy and straightforward to design a suitable parser should there be more details known about the bibliography editing style.

Listing 1: Bibliography reference parsing

```
1   dot = Literal(".")
2   comma = Literal(",")
3   alphas = ppu.Latin1.alphas+ppu.LatinA.alphas+ppu.LatinB.alphas
4   family_name = Word(alphas+"-", min=2)
5   init_name = Char(alphas)+dot+Optional("-"+Char(alphas)+dot)
```

```
6    same = Word("-")+dot.suppress()
7    year_or_range = r"\d{4}[a-z]?([,-]\d{4})?"
8    year = Regex(year_or_range)+Optional(dot|comma).suppress()
9    author = family_name("LastName")+comma+OneOrMore(init_name("FirstName"))
10   author_list = Combine((author|same)+Optional(dot|comma).suppress()
11   bib_reference = author_list("author")+year("year")+restOfLine("title")
```

In our generic implementation, we rely on a similar-style pattern to parse bibliography entries in the CMS Author-Date format, with some technical details to account for abbreviations like *eds.* (editors), and on a simple heuristic to parse entries not accepted by such a strict grammar. The latter attempts to find a year or a year range (using a regular expression from line 7 in Listing 1) anywhere in the text. It also assumes that the list of authors is always in front of the reference text, and that the work's title is the longest sentence between a given set of separators, such as dots or quotation marks.

3.1.2 *Parsing indexes*

As our examples in Section II illustrated, indexes in AHSS publications provide very specialized information under the assumptions that the reader knows how to interpret their content. The extraction of this information requires formal grammars for various index types and various conventions a publisher and/or authors choose to use. In our case study corpus, we observed the repeated use of certain patterns for certain index types, and developed a set of grammars for them. For example, the majority of index locorum entries are composed of a *label* consisting of one or more words, followed by a *loci* which are numeric expressions (possibly, with several levels separated by dots or dashes), and a set of occurrences in the current work, i.e., page numbers. Listing 2 provides a sample grammar for parsing strings that would accept index entries like:

> Agamemnon 42–4 591, 593
> Aeschylus Agamemnon 203–4/216–17 433
> Aeschylus Agamemnon 6–7 19 14 586 22 232, 410, 619 32 ff. 129
> Aeschines 2.157 291

Listing 2: Index parsing
```
1    alphas = ppu.Latin1.alphas+ppu.LatinA.alphas
2     +ppu.LatinB.alphas+ppu.Greek.alphas+"\"'.-_:&()/?"
3    ocr = delimitedList(Word(ppu.Latin1.nums))
4    locus_fragment = Word(ppu.Latin1.nums+".=")+Optional(oneOf("ff."))
5    locus = locus_fragment+Optional("/"+locus_fragment)
6    label = OneOrMore(Word(alphas+",;"))
7    index = label("label")+OneOrMore(locus("locus")+ocr("occurrences"))
```

A variation of the above pattern without *loci* part would accept index entries typical for indexes of names and subjects. In practice, one has to design such grammars given a complete list of syntactic rules and special keywords. On the plus side, the existing grammar specification libraries like the one used in our tool make it easy to design on-demand parsing algorithms and pass them into the processing pipeline whenever it is applied to a certain dataset.

3.2 Disambiguation

Disambiguation refers to the removal of ambiguity to narrow down the meaning of words or distinguishing between similar things (names, locations, subjects, etc.). In the context of our application, disambiguation is also an instrument to determine whether differently phrased bib-

liographic references imply the same published work, and whether index entries refer to the same term or concept despite some syntactic mismatch in their labels.

To be able to recognize that two or more bibliographic references imply the same work, we apply a simple clustering method based on matching of reference titles and publication years. References to the same work do not contain any term rearrangement and hence can be compared using a string-based similarity metric [Prasetya et al., 2018]. To compensate for possible typographic differences in reference titles due to the way different authors format references (e.g., with or without quotation marks) and because our parsing method cannot exactly separate titles from extra information provided in the extracted reference text, we rely on a fuzzy matching of reference titles by computing the Levenshtein editing distance [Levenshtein, 1965] between them. Two references are clustered together if their Levenshtein ratio is over a given threshold and publication years match. In this case, we create an instance of the class `Cluster` (see Section IV) which integrates similar references. Similarly, we can apply a clustering method based on the fuzzy string matching to index labels. However, index terms are much shorter, so clustering only makes sense if the parsing method performs well and the similarity threshold is high. We found clustering useful for indexes that refer to specialized information (e.g., index of ancient author names). In particular, clustering helps to reduce the number of requests to external services when we attempt to link extracted terms with their canonical representation in external databases.

To enable large-scale automatic bibliographic reference and index term disambiguation, we are interested in services with public APIs (Application Programming Interfaces). API is a protocol that allows a user to query a resource and retrieve data in a machine-readable format. A number of APIs that provide access to collections of published works such as books or scholarly journal articles are available to researchers. Some of them are open to the public, while others are available exclusively to libraries or require subscriptions. Due to the need to disambiguate a large number of extracted references, we are also interested in APIs without rigid constraints on the number of requests. Two global open APIs with unrestricted number of requests that enable search over published works are:

- Google Books API, in particular, the `Volume` collection GoogleBooksAPI, which shares metadata with industry identifiers (DOIs, ISBN) for books or magazines hosted by Google Books.
- Crossref API CrossrefAPI which provides metadata with DOIs for 100 million scholarly works.

In our pipeline, we disambiguate extracted references by executing HTTP requests to these services that search for works with best matching titles to ours. The responses are given in the form of a set of JSON objects from which we extract the key parameters of the reference, including the title and the publication year. We then check whether (i) the syntactic match between the returned title and our original title exceeds the similarity threshold and whether (ii) the publication years match. If both conditions hold, we create an instance of the `ExternalPublication` class (see Section IV) which unambiguously identifies the cited work (via industry identifiers and/or link to the metadata provided by the aforementioned APIs). The experimental evaluation of this process is discussed in Section V.

Similarly, we disambiguate index terms using two Knowledge Bases (KB):

- Wikidata, a free and open KB that acts as central storage for the structured data of projects such as Wikipedia (free encyclopedia), Wikisource (free library), Wiktionary (open dictionary of terms), and others. We rely on its API functionality that enables search for

entities using labels and aliases WikidataAPI.

- HuCit KB[5] [Romanello and Pasin, 2017] of classical (Greek and Latin) texts, developed with the aim of supporting the automatic extraction of bibliographic references to such texts. As a specialized KB, this resource is particularly useful for the disambiguation of author names and work titles, including their abbreviated versions, that occur in the indexes of ancient names and cited passages.

We evaluate bibliographic reference and index term disambiguation using the aforementioned methods in Section V.

IV DATA MODEL AND KNOWLEDGE GRAPH

4.1 Data model

The class diagram in Figure 5 shows our model for the representation of extracted data in preparation for the KG construction. As mentioned in Section III, to manage large datasets or publication archives, we split them into batches of a manageable size. All pipeline operations are performed on a single batch, and objects of the `Batch` class are created to represent the extracted data. Each batch records the information about the location of the dataset, its starting point (i.e., list index) and size, as well as a list of publications included to the batch, and, optionally, if the clustering is performed, references to cluster sets that group extracted bibliographic references and index terms.

The publication metadata is accumulated in the `BasePublication` class, which is a common class to define publications from a publisher's archive and external publications added at the bibliographic reference disambiguation step. The base publication object typically would include title, publication year, language, publisher, authors, editors, and industry identifiers. The object of the derived class `Publication` that represents a publication from the internal dataset additionally records the corresponding location (archive or data folder), relevant back matter file paths, i.e., JATS (BITS) file, PDF bibliography and PDF index files, as well as the lists of extracted bibliographic and index references.

The latter are represented by instances of the class `Reference` and `IndexReference`, respectively. Both classes extend the abstract class `BaseReference` that keeps the entity text and its order number in the publication's list of references or index terms. The specialization of these classes is in their ability to parse the specific entry text and produce structured representation of either a bibliographic reference or an index term. The former consists of the author, publication year, and title. It can also keep an optional pointer to a preceding reference to be able to derive author names from it if they are omitted in the reference text. The latter represents an index entry as a set of index parts which includes a label, (optional) locus, and a list of label occurrences (i.e., page numbers). Finally, both types of objects can refer to external resources that disambiguate them, which are modelled by the classes `ExternalPublication` and `ExternalIndex`, respectively. These classes keep links (URIs) to the public resources and the type of the API that supplied them (i.e., Google Books, Crossref, Wikidata, or Hucit). Additionally, to be able to execute meaningful queries over the KG, it is useful to copy some structured data from external objects to our model (e.g., external publication objects provided by Google Books or Crossref APIs allow us to get full contributor names, making it possible, for example, to discover other works of the same authors).

---

[5]https://pypi.org/project/hucitlib.

4.2 Knowledge graph

We store the data extracted in the processing pipeline in the form of nodes and relationships using the Neo4j graph database. Node labels in this graph correspond to the classes described above, with the exception of abstract classes and batch objects as they merely serve as helpers in data extraction processes. The relationships originate from the association links between the objects.

Figure 6 shows a KG fragment that displays a `Publication` node for the book with DOI 'B9789004339460' (randomly chosen from the Brill corpus to illustrate our data extraction process), along with a set of relevant index terms represented by the `IndexReference` nodes, with the edges (relations, labelled `Includes`) between them. The general pipeline discussed in Section III discovered the index file in the compressed publication folder, extracted and parsed 66 index entries, out of which 17 were disambiguated via the Wikidata API. The simple generic index parser we designed failed to parse 25 entries due to the use of the specialized characters from the Brill typeface. The external URLs provided by the Wikidata API spawned the `ExternalIndex` nodes, connected to the corresponding index nodes via the KG edges labelled as `RefersTo`. For example, the selected node corresponds to the entry 'Lupieri, Edmondo 9, 80' and resolves to the Wikidata term 'Q85317538'. [6]

Figure 7 shows a KG fragment that displays a chain linking the same `Publication` node with a `Reference` node, via the edge labelled `Cites`. The reference node originates from the following text [7]:

> Adam, Alfred. 1959. Die Psalmen des Thomas und das Perlenlied als Zeugnisse vorchristlicher Gnosis, Berlin: Alfred Töpelmann.

The next node in the line is the `ExternalPublication` node that disambiguates this reference via the Crossref API. Finally, the last shown node is the `Contributor` node that provides the name and surname of the cited work's author, extracted from the Crossref's record.

4.3 Implementation and dissemination

The source code for our information extraction and KG construction software is openly available [8]. The Python package provides a proof-of-concept implementation of the described pipeline (see Section III) that expects as input a path to a folder with compressed publication archive (see Section 2.3) and a URI to the Neo4j database instance to store the KG. Besides the implementation of the data model (see Section 4.1), the toolset includes a `DBConnector` class which provides CRUD (Create, Read, Update, Delete) operations on the KG and useful methods for data refinement such as cluster or external link node merging. These operations can be exposed via a REST API to other systems to retrieve or manipulate the acquired data, in particular, enable useful queries for data analysis (which, however, is out of the scope of the current project). We also illustrate, via a series of tests, how the pipeline can be customized with different parsers and disambiguation services, and experiment with alternative serialization methods such as translation of our data model (see Section 4.1) into formats defined by SPAR (Semantic Publishing and Referencing) ontologies. The SPAR ontologies form a suite of modules for the creation of comprehensive machine-readable RDF (Resource Description Framework) metadata for every aspect of semantic publishing and referencing such as document description, biblio-

---

[6]From: https://brill.com/view/book/9789004339460/B9789004339460_018.xml.

[7]Extracted from the PDF bibliography file at https://brill.com/view/book/9789004339460/B9789004339460_017.xml.

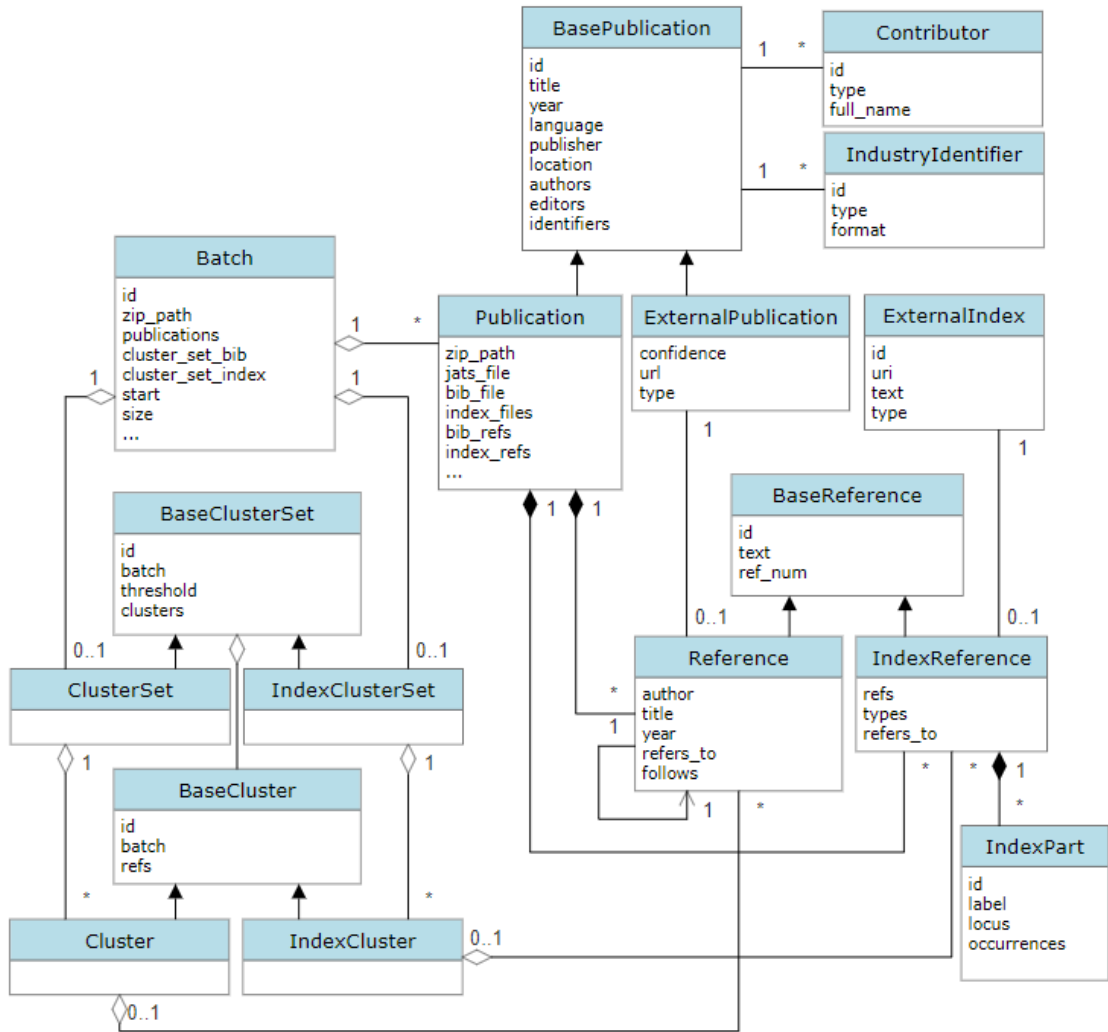[8]https://github.com/kiem-group/pdfParser.

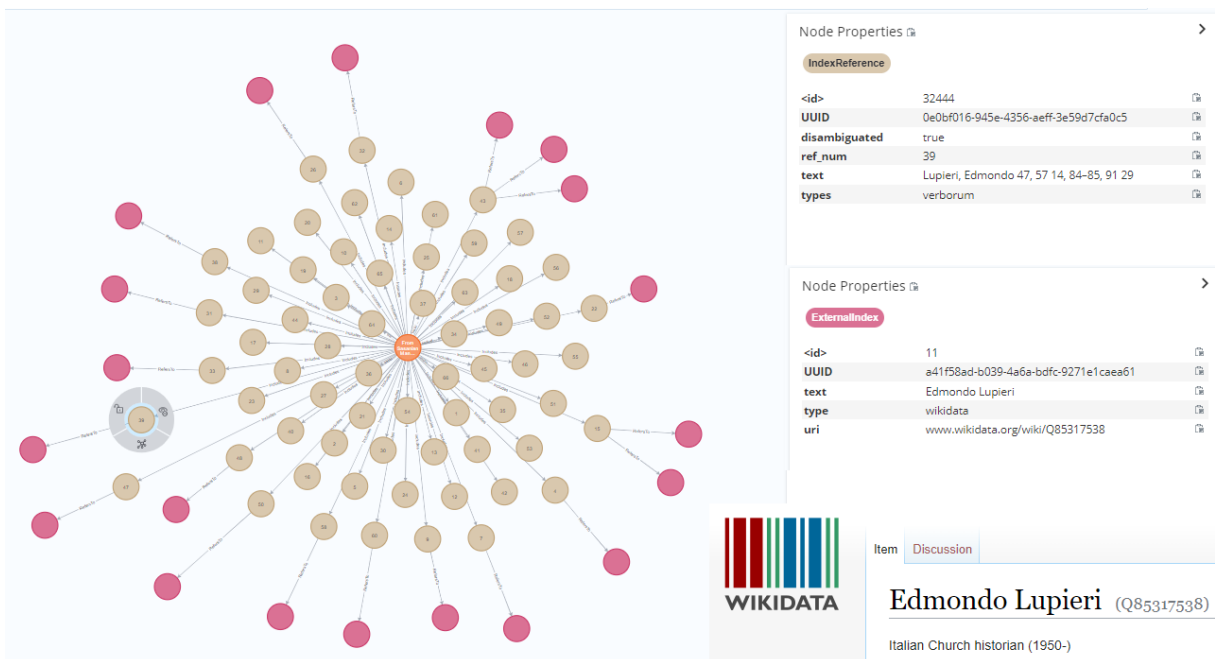Figure 5: Class diagram of the PDF Parser data model.

Figure 6: Publication node with related index terms and links.

Figure 7: Bibliographic reference node with disambiguation link.

graphic resource identifiers, types of citations and related contexts, bibliographic references, document parts, agents' roles and contributions, and more [Peroni and Shotton, 2018]. We enable partial mapping of our data model into SPAR RDF. Listing 3 illustrates how a publication with corresponding automatically extracted bibliographic references can be represented in the OpenCitations Data Model (OCDM) [Daquino et al., 2020].

Listing 3: SPAR bibliographic references

```
1    @prefix : <http://brill.com/kiem/> .
2    @prefix biro: <http://purl.org/spar/biro> .
3    @prefix co: <http://purl.org/co/> .
4    @prefix frbr: <http://purl.org/vocab/frbr/core#> .
5    @prefix c4o: <http://purl.org/spar/c4o> .
6    <http://dx.doi.org/10.1163/9789004261648> frbr:part :reference-list .
7    :ref-001 a biro:BibliographicReference ;
8        c4o:hasContent "Augustine Confessiones. Edited by Martin Skutella.
9        Leipzig: Teubner, 1934. " .
10   :ref-002 a biro:BibliographicReference ;
11       c4o:hasContent "Chaldaean Oracles Confessions. Translated by Henry
12       Chadwick. Oxford: Oxford University " .
13   ...
```

The OCDM is the most notable realization of the SPAR general guidelines used for storing data in OpenCitations datasets [Peroni and Shotton, 2020]. One of such datasets, COCI [Heibi et al., 2019], the OpenCitations Index of Crossref open DOI-to-DOI citations, is an RDF dataset containing details of all the citations that are specified by the open references to DOI-identified works present in Crossref. Table 1 compares the number of extracted and disambiguated references for five randomly chosen publications from the Brill catalogue with the number of DOI-to-DOI citations provided by the COCI public API [9]. The significant gap between the number of actually cited works in a book, or even between the number of works we were able to disambiguate (associate with their unique identifiers, DOI and/or ISBN, via Crossref and Google Books APIs), suggests that our tool could be employed by other agents, publishers or authors, for augmenting existing datasets with new relationships.

---

[9] http://opencitations.net/index/coci/api/v1.

# refs / DOI	9789004180994	9789004189829	9789004231283	9789004257788	9789004277151
Extracted	335	325	365	366	220
Disambiguated	145	96	219	138	94
COCI	9	40	73	6	7

Table 1: Number of extracted references vs number of COCI DOI-to-DOI citations.

V   EVALUATION

In this section, we reflect on the quality of extracted data by providing some experimental evaluation of the involved pipeline steps.

5.1   Quality of PDF parsing

The first critical step in data extraction process is splitting the back matter file into individual bibliographic and index references, i.e., a heuristic method relying on the alignment of reference text in a PDF file (see Section 3.1). One way to evaluate the quality of this method would be to compare the extracted reference text with "ground truth" reference text in a labelled dataset. We employed this method in unit tests on selected references, but, unfortunately, we did not have access to a sufficiently large dataset of this kind. Nevertheless, the evaluation of this step is subsumed to the evaluation of the disambiguation steps as the human curator evaluating the correctness of external links could also judge the quality of the randomly selected references. Out of a sample of 500 text fragments assumed to be bibliographic references, 495 were valid and complete references, and

- 2 entries contained other type of information; they were dismissed by the bibliographic reference parser as no publication year was found in these fragments;
- 3 entries contained incomplete reference text: 2 were lacking the tail part yet contained enough information to identify the referred works, and 1 lacked the head part containing authors and publication year.

The aforementioned observations make us confident that our layout-based reference extraction method works very well given a PDF file with list of references, demonstrating almost 99% accuracy on randomly selected subsets from Brill's corpus.

The risk of KG contamination still cannot be excluded if the same approach was to be applied to a PDF containing other type of data (e.g., book body chapter) or unusually structured bibliography file that alternates references with paragraphs of other content. Hence, we parsed only back matter files which we were sure contain the right information based on the presence of keywords "bibliography" and "index" in their titles. Given this constraint, in the Brill's corpus of 1899 publications, the pipeline processed 650 bibliography and 1804 index files.

5.2   Quality of clustering

For the evaluation of the clustering method we used a ground truth dataset [Romanello, 2022], consisting of 3579 pairs of bibliographic references. For each labelled pair, a manually assigned score indicates whether the two references are referring to the same bibliographic entity or not (1.0 = true, 0.0 = false, 0.5 = partly), e.g.:

> C. Lane, Venise, une République maritime, Paris, 1988, p. 344;
> Lane, F.: Venise, une république maritime. Paris 1985. p. 69.
> Score: 0.0
> ...
> C. Lane, Venise, une République maritime, Paris, 1988, p. 344;
> Lane, Frédéric Chapin. Venise: une république maritime, préface de Fernand Braudel; trail,
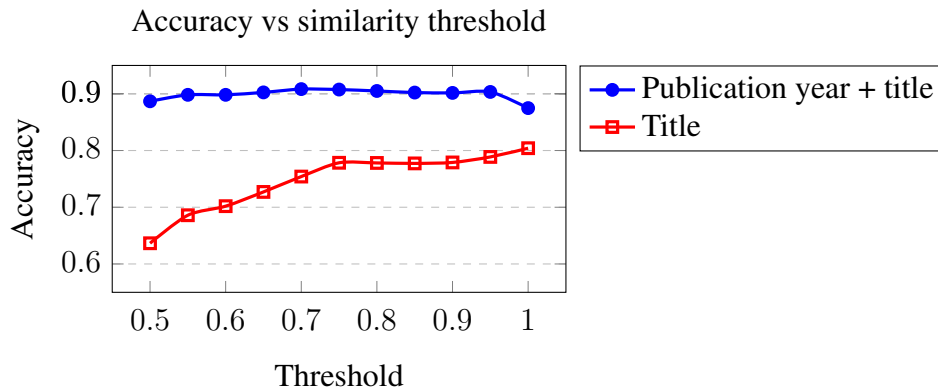
Figure 8: Evaluation of the bibliographic reference clustering.

de l'américain par Yannick Bourdoiseau et Marie Ymonel. Paris, Flammarion. 1988.
Score: 1.0

We parsed the given pairs of text references and computed the Levenshtein similarity ratio between their titles (more specifically, the shortest title and the part of the longer title of the same length). Two references are clustered together if their publication years match and the similarity score exceeds a given threshold. The first plot in Figure 8 shows the accuracy of the similarity-based clustering for various thresholds, reaching 90.8% accuracy for the value of 0.7.

The close accuracy values for various thresholds indicate that the decision in this artificial dataset significantly depends on the matching of the publication years. Indeed, by matching publication years alone, we yield the 82% accuracy. Obviously, in large publication archives, title matching will have the primary role in clustering references to the same work. To choose the best similarity threshold, we therefore repeated this experiment matching publication titles alone. The second plot in Figure 8 shows that, for the threshold values above 0.75, the accuracy is close to 80%.

Based on this evaluation, we chose the default threshold of 0.75 for bibliographic reference title matching in disambiguation methods of our pipeline.

5.3 Quality of disambiguation

To evaluate the quality of disambiguation step, we randomly selected 500 extracted references from Brill's corpus, and generated a spreadsheet that associates the reference text with URLs supplied by the Google Books and Crossref APIs in response to the disambiguation requests in the pipeline. We then asked a human expert, a Brill's employee, to evaluate whether the URLs refer to the correct work, with score being 0 for a wrong or absent link, 1 for the correct link, and 0.5 for a partially correct link, i.e., a URL of the book collection containing the cited work or a different edition of the cited work. Moreover, for entries with incorrect or missing links, the expert was asked to manually locate publicly available link matching the work in the reference. Similarly, a spreadsheet containing 500 randomly selected index references was generated together with Wikidata responses disambiguating labels from these entries. In our early experiments, we observed that the Wikidata API [10] often does not produce results for queries containing author surname followed by the comma-separated name(s) as opposed to the name(s) followed by the surname (e.g., "Gournay, Marie le Jars de" vs "Marie le Jars de Gournay"). Adding an extra query with transformed search label for indexes with a comma after

---

[10]https://www.wikidata.org/w/api.php?action=wbsearchentities&search.

the first term helped us to compensate for this effect and acquire identifiers for more entries.

Our disambiguation method resolved 209 entries for 500 bibliographic references, and the human expert assigned the total score of 169 points to their correctness; this allows us to estimate its precision as $169/209$, or 80.8%. At the same time, the human expert was able to find 149 extra links in addition to the 209 we filled in automatically, increasing the number of disambiguated references to 358. On this ground, we estimate the recall of our method as $209/358$ or 58.4%. The analogous approach for the index disambiguation yielded the score of 259 for the total of 274 automatically discovered Wikidata identifiers, setting precision to 94.5%. With the overall total of 414 links, automatically and manually located, the recall is estimated to be at 66%. [11]

We did not evaluate Hucit-based disambiguation in the same style since it is a highly specialized database that searches for exact terms and, considering that we do not retain context information, any response, if present, is likely correct. The coverage hugely depends on the index content and hence is not representative either. Moreover, its API generates much slower responses, so this disambiguation service is recommended for targeted disambiguation, when a human supervisor considers its use more promising than the generic Wikidata search.

VI   RELATED WORK

Our work relates to three distinct areas of research: a) the construction of knowledge graphs, b) the extraction of bibliographic information from AHSS publications and c) the processing of indexes.

6.1   Knowledge Graphs

The development of KGs in the publishing sector is part of the ongoing transformation of publications from document-centred objects to structured, interlinked knowledge representations. KGs such as Microsoft Academic Knowledge Graph [Färber, 2019], SciGraph [Yaman et al., 2019], the Literature Graph [Ammar et al., 2018] and the Semantic Scholar Open Research Corpus [Lo et al., 2020] underpin the search functionalities of academic search engines like Google Scholar, Microsoft Academic and Semantic Scholar. AHSS disciplines, however, tend to be underrepresented and poorly covered in such KGs which are developed with a focus on STM disciplines. Within this landscape, the open KG of Wikidata represents quite an exception as several resources in the Humanities (e.g., gazetteers, catalogues) have been linked to entries in Wikidata, thus making it a useful resource when extracting and linking structured knowledge from publications in this area [Haslhofer et al., 2018]. The High Integration of Research Monographs in the European Open Science (HIRMEOS) project Bertino [2017] is to address the particularities of academic monographs in the AHSS domain and foster integration of monographs into the European Open Science Cloud.

6.2   Citation Mining

In this paper, we approached the extraction of bibliographic references by parsing the bibliography section of each book, after having it located by applying some heuristics on the available book-level metadata. This approach, however, is not viable when structural metadata about the publication are lacking, or when references are not grouped in a dedicated section (such is the case with journal articles). An alternative to the parsing of bibliography sections are end-to-end

---

[11]The complete evaluation tables are available at https://github.com/kiem-group/pdfParser/tree/main/data_test.

approaches which locate references within the full-text of publications, and successively segment them into their components. Existing services that implement such end-to-end extraction of references are GROBID [Lopez, 2009], CERMINE [Tkaczyk et al., 2015], BILBO [Kim et al., 2011] and EXCITE [Hosseini et al., 2019], with the last two having a specific focus on AHSS publications. While these services are able to recognize references in mixed documents, their evaluation on our sample PDF documents containing exclusively bibliographic references revealed that the basic heuristic-based reference splitting procedure we proposed performs significantly better.

While the large majority of work in this area dealt with references to secondary literature, the extraction of references to primary sources was investigated, in particular with respect to archival references [Rodrigues Alves et al., 2018] and canonical references to Greek and Latin literary works [Romanello, 2015].

6.3 Index Processing

When it comes to the processing of indexes, we should distinguish between a) processing aimed at the semi-automatic creation of indexes and b) processing existing indexes (i.e., parsing) with the aim of exploiting the structured information they contain.

With respect to the creation of indexes, the landscape is dominated by commercial indexing software (e.g., CINDEX, Macrex, Sky Index) that automate certain indexing-related tasks – such as sorting, formatting, cross-referencing – while the human in the loop is still responsible for the main intellectual work entailed by creation of an index. Romanello [2019], in collaboration with the publisher De Gruyter, has recently showed how the creation of an index locorum containing several thousands of entries can be largely automated by using open citation mining software.

Researchers, however, have also recognized the value of existing indexes as sources of structured, human-curated information. Recent work by Blidstein and Zhitomirsky-Geffet [2022] provides a fitting example of how indexes, and *indices locorum* in particular – can provide extremely valuable data for citation analysis. By parsing such indexes, they distilled information about cited primary sources within a corpus of scholarly books on ancient Mediterranean religion and culture. They were able to explore how various network types capture distinct aspects of published scholarship thanks to this corpus, which also allowed for systematic analysis and comparison of various citation network types. Furthermore, parsing of digitized indexes was employed for the construction of controlled vocabularies [Piotrowski and Senn, 2012], gazetteers of historical places [Piotrowski, 2010] and literary authors and works [Romanello et al., 2009], for improving access to historical archives [Colavizza et al., 2019], for building character networks from literary texts [Rochat, 2014], and for reconstructing the ontological relations implied by hierarchical relations between index terms [Li et al., 2019].

VII CONCLUSION

In this work, we proposed an open and collaborative approach to facilitating the indexation of AHSS literature in modern scientific search engines. We openly released a software implementing a pipeline able to extract structured information from the back matter of books, specifically references and indexes. The pipeline further implements disambiguation and normalization routines on the extracted information, and constructs a knowledge graph using the SPAR and OpenCitations data models. Our work is aimed at supporting small and medium enterprise (SME) publishers in the arts, humanities and social sciences (AHSS). We seek to provide them

with a means of contributing their own publication knowledge graphs in the open domain. To exemplify and evaluate the proposed pipeline, we applied it on Brill's Classics corpus, with promising results.

We identify several directions for future work. Firstly, our proposed pipeline implementation can be improved in a variety of ways, for example by expanding the coverage of data formats and contents it can handle, and the methods used for PDF parsing, clustering or entity disambiguation. We believe this will happen only if the pipeline is adopted by a community of users, primarily AHSS SME publishers. This is why another important direction for future work includes the dissemination of our project results and the maintenance of the software we released. Lastly, more effort is needed to fully integrate the proposed pipeline within the existing and growing open science ecosystem. Making sure that the extracted information is linked to mainstream open authority records, as well as ingested in OpenCitations is of critical importance to the success of the proposed approach. Our contribution here made the first, but not yet all the steps in this direction.

Ideally, in the future we hope to see a wide adoption of the approach we propose by AHSS publishers, so that the software we released will become a live codebase with a diverse set of users and contributors. We believe that, with an open source, collaborative approach, we can significantly improve the indexing and discoverability of AHSS literature.

CODE AND DATA AVAILABILITY

The source code of the proposed information extraction and knowledge graph construction software is openly available: https://github.com/kiem-group/pdfParser.

ACKNOWLEDGEMENTS

References

Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu A. Ha, Rodney Michael Kinney, Sebastian Kohlmeier, Kyle Lo, Tyler C. Murray, Hsu-Han Ooi, Matthew E. Peters, Joanna L. Power, Sam Skjonsberg, Lucy Lu Wang, Christopher Wilhelm, Zheng Yuan, Madeleine van Zuylen, and Oren Etzioni. Construction of the Literature Graph in Semantic Scholar. In *The North American Chapter of the Association for Computational Linguistics (NAACL)*, volume 3, pages 84–91. ACL, 2018. doi: 10.18653/v1/N18-3011.

Andrea Bertino. Hirmeos high integration of research monographs in the european open science infrastructure. *Septentrio Conference Series*, 11 2017. doi: 10.7557/5.4275.

BITS. Book Interchange Tag Suite (BITS), version 2.0, 2016. URL https://jats.nlm.nih.gov/extensions/bits/tag-library/2.0/index.html.

Moshe Blidstein and Maayan Zhitomirsky-Geffet. Towards a new generic framework for citation network generation and analysis in the humanities. *Scientometrics*, 127(7):4275–4297, July 2022. ISSN 1588-2861. doi: 10.1007/s11192-022-04438-y.

BrillTypeface. Brill Typeface. Extensive version 4.0 Upgrade (2021, 2021. URL https://brill.com/page/BrillFont/brill-typeface.

Cosetta Cadau. *Studies in Colluthus' Abduction of Helen*. Brill, 2015. ISBN 978-90-04-28959-8. doi: https://doi.org/10.1163/9789004289598. URL https://brill.com/view/title/26617.

Giovanni Colavizza and Matteo Romanello. Citation Mining of Humanities Journals: The Progress to Date and the Challenges Ahead. *Journal of European Periodical Studies*, 4(1):36–53, 2019. ISSN 2506-6587. doi: 10.21825/jeps.v4i1.10120. URL https://openjournals.ugent.be/jeps/article/id/71493/.

Giovanni Colavizza, Matteo Romanello, and Frédéric Kaplan. The references of references: a method to enrich humanities library catalogs with citation data. *International Journal on Digital Libraries*, 19(2-3):151–161, 2018. ISSN 1432-5012, 1432-1300. doi: 10.1007/s00799-017-0210-1. URL http://link.springer.com/10.1007/s00799-017-0210-1.

Giovanni Colavizza, Maud Ehrmann, and Fabio Bortoluzzi. Index-Driven Digitization and Indexation of Historical Archives. *Frontiers in Digital Humanities*, 6, 2019. ISSN 2297-2668.

Giovanni Colavizza, Silvio Peroni, and Matteo Romanello. The case for the Humanities Citation Index (HuCI): a citation index by the humanities, for the humanities. *ArXiv*, abs/2110.00307, 2021. doi: 10.48550/ARXIV.2110.00307. URL https://arxiv.org/abs/2110.00307.

CrossrefAPI. Crossref REST API Documentation, 2016. URL https://www.crossref.org/documentation/retrieve-metadata/rest-api/.

CWKB. Classical Works Knowledge Base (CWKB), 2020. URL http://cwkb.org/.

Marilena Daquino, Silvio Peroni, David Shotton, Giovanni Colavizza, Behnam Ghavimi, Anne Lauscher, Philipp Mayr, Matteo Romanello, and Philipp Zumstein. The OpenCitations Data Model. In Jeff Z. Pan, Valentina Tamma, Claudia d'Amato, Krzysztof Janowicz, Bo Fu, Axel Polleres, Oshani Seneviratne, and Lalana Kagal, editors, *The Semantic Web ISWC 2020*, volume 12507, pages 447–463. Springer, Cham, 2020. ISBN 978-3-030-62465-1 978-3-030-62466-8. doi: 10.1007/978-3-030-62466-8_28. URL https://link.springer.com/10.1007/978-3-030-62466-8_28. LNCS.

Michael Färber. The Microsoft Academic Knowledge Graph: A Linked Data Source with 8 Billion Triples of Scholarly Data. In Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtěch Svátek, Isabel Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon, editors, *The Semantic Web – ISWC 2019*, volume 11779, pages 113–129. Springer, Cham, 2019. ISBN 978-3-030-30795-0 978-3-030-30796-7. doi: 10.1007/978-3-030-30796-7_8.

GoogleBooksAPI. Google Books APIs: Volume, 2012. URL https://developers.google.com/books/docs/v1/reference/volumes.

Bernhard Haslhofer, Antoine Isaac, and Rainer Simon. Knowledge Graphs in the Libraries and Digital Humanities Domain. *arXiv:1803.03198 [cs]*, pages 1–8, 2018. doi: 10.1007/978-3-319-63962-8_291-1.

Ivan Heibi, Silvio Peroni, and David Shotton. Software review: COCI, the OpenCitations Index of Crossref open DOI-to-DOI citations. *Scientometrics*, 121, 2019. doi: 10.1007/s11192-019-03217-6.

Azam Hosseini, Behnam Ghavimi, Zeyd Boukhers, and Philipp Mayr. EXCITE – A Toolchain to Extract, Match and Publish Open Literature References. In *2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 432–433, 2019. doi: 10.1109/JCDL.2019.00105.

Mohamad Yaser Jaradeh, Allard Oelen, Kheir Eddine Farfar, Manuel Prinz, Jennifer D'Souza, Gábor Kismihók, Markus Stocker, and Sören Auer. Open Research Knowledge Graph: Next Generation Infrastructure for Semantic Scholarly Knowledge. In *Proceedings of the 10th International Conference on Knowledge Capture*, pages 243–246, Marina Del Rey CA USA, 2019. ACM. ISBN 978-1-4503-7008-0. doi: 10.1145/3360901.3364435. URL https://dl.acm.org/doi/10.1145/3360901.3364435.

JATS. JATS: Journal Article Tag Suite, version 1.3, 2021. URL http://www.niso.org/publications/z3996-2021-jats.

Young-Min Kim, Patrice Bellot, Elodie Faath, and Marin Dacos. Automatic annotation of bibliographical references in digital humanities books, articles and blogs. In *4th ACM Workshop on Online Books, Complementary Social Media and Crowdsourcing - BooksOnline '11*, Glasgow, United Kingdom, 2011. ACM. doi: 10.1145/2064058.2064068.

Emanuel Kulczycki, Tim C. E. Engels, Janne Plnen, Kasper Bruun, Marta Dušková, Raf Guns, Robert Nowotniak, Michal Petr, Gunnar Sivertsen, Andreja Istenič Starčič, and Alesia Zuccala. Publication patterns in the social sciences and humanities: evidence from eight European countries. *Scientometrics*, 116(1):463–486, 2018. ISSN 0138-9130, 1588-2861. doi: 10.1007/s11192-018-2711-0. URL http://link.springer.com/10.1007/s11192-018-2711-0.

Emanuel Kulczycki, Raf Guns, Janne Pölönen, Tim C. E. Engels, Ewa A. Rozkosz, Alesia A. Zuccala, Kasper Bruun, Olli Eskola, Andreja Istenič Starčič, Michal Petr, and Gunnar Sivertsen. Multilingual publishing in the social sciences and humanities: A sevencountry European study. *Journal of the Association for Information Science and Technology*, 71(11):1371–1385, 2020. ISSN 2330-1635, 2330-1643. doi: 10.1002/asi.24336. URL https://onlinelibrary.wiley.com/doi/10.1002/asi.24336.

Vladimir I. Levenshtein. Binary Codes Capable of Correcting Spurious Insertions and Deletions of Ones. *Probl. Inf. Transm.*, 1(1):8–17, 1965.

Ning Li, Meng Tian, and Shuqi Lv. Extracting Hierarchical Relations Between the Back-of-the-Book Index Terms. In *Chinese Lexical Semantics: 20th Workshop, CLSW 2019, Beijing, China, June 28–30, 2019, Revised Selected Papers*, pages 433–443, Berlin, Heidelberg, 2019. Springer-Verlag. ISBN 978-3-030-38188-2. doi: 10.1007/978-3-030-38189-9_45.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. S2ORC: The Semantic Scholar Open Research Corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983. ACL, 2020. doi: 10.18653/v1/2020.acl-main.447.

Patrice Lopez. GROBID: Combining Automatic Bibliographic Data Recognition and Term Extraction for Scholarship Publications. In Maristella Agosti, José Borbinha, Sarantos Kapidakis, Christos Papatheodorou, and Giannis Tsakonas, editors, *Research and Advanced Technology for Digital Libraries*, Lecture Notes in Computer Science, pages 473–474, Berlin, Heidelberg, 2009. Springer. ISBN 978-3-642-04346-8. doi: 10.1007/978-3-642-04346-8_62.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. Multi-Task Identification of Entities, Relations, and Coreference for Scientific Knowledge Graph Construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium, 2018. ACL. doi: 10.18653/v1/D18-1360. URL http://aclweb.org/anthology/D18-1360.

Alberto Martín-Martín, Mike Thelwall, Enrique Orduna-Malea, and Emilio Delgado López-Cózar. Google Scholar, Microsoft Academic, Scopus, Dimensions, Web of Science, and OpenCitations COCI: a multidisciplinary comparison of coverage via citations. *Scientometrics*, 126(1):871–906, 2021. ISSN 0138-9130, 1588-2861. doi: 10.1007/s11192-020-03690-4. URL https://link.springer.com/10.1007/s11192-020-03690-4.

PDFMiner. PDFMiner: Python PDF parser and analyzer, 2019. URL https://pypi.org/project/pdfminer/.

Silvio Peroni and David Shotton. The SPAR Ontologies. In Denny Vrandečić, Kalina Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee, and Elena Simperl, editors, *The Semantic Web ISWC 2018*, volume 11137, pages 119–136. Springer, Cham, 2018. ISBN 978-3-030-00667-9 978-3-030-00668-6. doi: 10.1007/978-3-030-00668-6_8. URL http://link.springer.com/10.1007/978-3-030-00668-6_8. LNCS.

Silvio Peroni and David Shotton. OpenCitations, an infrastructure organization for open scholarship. *Quantitative Science Studies*, 1(1):428–444, 2020. ISSN 2641-3337. doi: 10.1162/qss_a_00023. URL https://direct.mit.edu/qss/article/1/1/428-444/15580.

Michael Piotrowski. Leveraging back-of-the-book indices to enable spatial browsing of a historical document collection. In *Proceedings of the 6th Workshop on Geographic Information Retrieval*, GIR '10, pages 1–2, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-826-1. doi: 10.1145/1722080.1722102.

Michael Piotrowski and Cathrin Senn. Harvesting indices to grow a controlled vocabulary: Towards improved access to historical legal texts. In *Proceedings of the 6th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, LaTeCH '12, pages 24–29, USA, 2012. ACL.

Didik Prasetya, Aji Wibawa, and Tsukasa Hirashima. The Performance of Text Similarity Algorithms. *International Journal of Advances in Intelligent Informatics*, 4, 2018. doi: 10.26555/ijain.v4i1.152.

PyParsing. PyParsing A Python Parsing Module, 2022. URL https://pypi.org/project/pyparsing/.

Yannick Rochat. *Character Networks and Centrality*. PhD thesis, University of Lausanne, 2014.

Danny Rodrigues Alves, Giovanni Colavizza, and Frédéric Kaplan. Deep Reference Mining From Scholarly Literature in the Arts and Humanities. *Frontiers in Research Metrics and Analytics*, 3, 2018. ISSN 2504-0537.

Matteo Romanello. *From Index Locorum to Citation Network: An Approavch to the Automatic Extraction of Canonical Reeferences and Its Applications to the Study of Classical Texts*. Doctoral thesis, King's College London, London, UK, 2015. URL https://dx.doi.org/11858/00-1780-0000-002A-4537-A.

Matteo Romanello. Experiments in digital publishing: creating a digital compendium. In Christiane Reitz and Simone Finkmann, editors, *Structures of Epic Poetry*, pages 331–348. De Gruyter, 2019. ISBN 978-3-11-049259-0. doi: 10.1515/9783110492590-074.

Matteo Romanello. Scholarindex/gt-reference-clustering: Version 1.0, September 2022. URL https://doi.org/10.5281/zenodo.7078762.

Matteo Romanello and Michele Pasin. Using linked open data to bootstrap a knowledge base of classical texts. In Alessandro Adamou, Enrico Daga, and Leif Isaksen, editors, *Proceedings of the Second Workshop on Humanities in the Semantic Web (WHiSe II)*, volume 2014 of *CEUR Workshop Proceedings*, pages 3–14. CEUR-WS.org, 2017. URL http://ceur-ws.org/Vol-2014/paper-01.pdf.

Matteo Romanello, Monica Berti, Alison Babeu, and Gregory Crane. When printed hypertexts go digital: Information extraction from the parsing of indices. In *Proceedings of the 20th ACM Conference on Hypertext and*

Hypermedia, HT '09, pages 357–358, New York, NY, USA, 2009. AACM. ISBN 978-1-60558-486-7. doi: 10.1145/1557914.1557987.

Dominika Tkaczyk, Paweł Szostek, Mateusz Fedoryszak, Piotr Jan Dendek, and Łukasz Bolikowski. CERMINE: Automatic extraction of structured metadata from scientific literature. *International Journal on Document Analysis and Recognition (IJDAR)*, 18(4):317–335, 2015. ISSN 1433-2825. doi: 10.1007/s10032-015-0249-8.

Sytze van der Veen. *Brill 325 years of scholarly publishing*, chapter Digital Times, page 163. Brill, 2008. URL https://scholarlyeditions.brill.com/reader/urn:cts:brillLit:brill.brill325.se-ed-eng01:163.

Jan van Waarden. Brill Typographic Style, 2020. URL https://confluence.brill.com/display/BE/Brill+Typographic+Style.

Martijn Visser, Nees Jan van Eck, and Ludo Waltman. Large-scale comparison of bibliographic data sources: Scopus, Web of Science, Dimensions, Crossref, and Microsoft Academic. *Quantitative Science Studies*, 2 (1):20–41, 2021. ISSN 2641-3337. doi: 10.1162/qss_a_00112. URL https://direct.mit.edu/qss/article/2/1/20/97574/Large-scale-comparison-of-bibliographic-data.

Jacob White. PubMed 2.0. *Medical Reference Services Quarterly*, 39(4):382–387, 2020. ISSN 0276-3869, 1540-9597. doi: 10.1080/02763869.2020.1826228. URL https://www.tandfonline.com/doi/full/10.1080/02763869.2020.1826228.

WikidataAPI. MediaWiki API help, 2012. URL https://www.wikidata.org/w/api.php?action=help&modules=wbsearchentities.

Beyza Yaman, Michele Pasin, and Markus Freudenberg. Interlinking SciGraph and DBpedia Datasets Using Link Discovery and Named Entity Recognition Techniques. In Maria Eskevich, Gerard de Melo, Christian Fäth, John P. McCrae, Paul Buitelaar, Christian Chiarcos, Bettina Klimek, and Milan Dojchinovski, editors, *2nd Conference on Language, Data and Knowledge (LDK 2019)*, volume 70 of *OpenAccess Series in Informatics (OASIcs)*, pages 1–8, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. ISBN 978-3-95977-105-4. doi: 10.4230/OASIcs.LDK.2019.15. URL http://drops.dagstuhl.de/opus/volltexte/2019/10379.