OXPath-based Data Acquisition for dblp

Christopher Michels
Trier University, dblp
michelsc@uni-trier.de

Ruslan R. Fayzrakhmanov
University of Oxford
ruslan.fayzrakhmanov@cs.ox.ac.uk

Michael Ley
Trier University, dblp
ley@uni-trier.de

Emanuel Sallinger
University of Oxford
emanuel.sallinger@cs.ox.ac.uk

Ralf Schenkel
Trier University, dblp
schenkel@uni-trier.de

ABSTRACT

We demonstrate how the contemporary problems of data acquisition for dblp can be tackled with OXPath. It enables web data extraction and wrapper maintenance for heterogeneous data sources on a simple declarative level. Its features render it a feasible instrument to retrieve the varying and changing web representations of the prototypical substructures in the bibliographic domain.

CCS CONCEPTS

•**Information systems** →**Digital libraries and archives;**

KEYWORDS

Bibliography, dblp, digital libraries, OXPath, web data extraction

ACM Reference format:
Christopher Michels, Ruslan R. Fayzrakhmanov, Michael Ley, Emanuel Sallinger, and Ralf Schenkel. 2017. OXPath-based Data Acquisition for dblp. In *Proceedings of Joint Conference on Digital Libraries, Toronto, Ontario, Canada, June 2017 (JCDL'17),* 2 pages.
DOI: 10.475/123_4

1 INTRODUCTION

With the popularity of the Semantic Web and the promotion of interoperability standards[1], publishers are ideally expected to provide access to structured data sources. However, most of them settle for electronic catalogs designed for human consumption. Thus, data acquisition requires simulated user interaction with sophisticated interfaces to query web documents. The increasing prevalence of JavaScript-driven web applications among the publishers of more than 1,500 journals and more than 4,000 conference and workshop series listed by the open bibliographic information provider *dblp computer science bibliography*[2] constitutes a problem of data harvesting. The dblp crawling architecture meets the resulting extent of data heterogeneity by leveraging more than 100 customized wrappers. These software components extract data from the HTML source of publisher websites based on regular expressions. This approach reaches its limits for two reasons: The wrappers are impractical or insufficient with the interactive elements of dynamic web content. Furthermore, they are expensive to maintain due to frequent layout changes in the source code of web pages.

Among other available extraction methods, the open source web data extraction language OXPath[3] [1] constitutes a more feasible solution to both problems: Its declarative expressions preserve the simplicity of the XML query language XPath, abstract from the implementation details of traditional dblp wrappers and enable more feasible wrapper maintenance. It extends XPath with action steps to simulate user interaction with web applications, the Kleene star for iteration, and markers for the extraction of a tree-like result across multiple interactions and visited pages. Its pliable output-handling interface suits the needs of differing recording guidelines among database publishers. With examples of OXPath-based wrappers for dblp, we demonstrate how OXPath tackles the deficits of the replaced dblp wrappers and synergizes well with the common, recurring substructures of the bibliographic domain.

2 EXTRACTING METADATA WITH OXPATH

Harvesting bibliographic metadata assumes various forms of extraction processes and involves crawling data on the Surface Web as well as leveraging complex web interfaces to pull relevant data from the Deep Web. The back-end databases of digital libraries are queried by submitting search terms to a form, applying filter options as necessary, and harvesting the resulting list of publications. For conference or workshop proceedings, articles usually are extracted from a plain table of contents. An overview page for all events of a series as a higher level of extraction is not always provided. The websites of journal publishers typically require a sequence of extraction steps for the following publication components: 1) volumes, 2) issues, and 3) articles. A generic OXPath wrapper considering all these relevant information levels can be represented by the following schematic expression.

```
1 doc('URL')
2   /volume_path:<volume>
3   [./{action}/issue_path:<issue>
4       [./{action}/(/paginator_path/{action})*
5         /article_path:<article>
6         [./{action}/data_path:<data>] ] ]
```

First, the expression visits the web page for a given journal. Then, it navigates, interacts, and extracts from the different structural levels involved. Interactions {action} can be related to clicking ({click}) or entering search terms into a text field ({"OXPath"})

[1] https://www.openarchives.org
[2] http://dblp.org
[3] http://www.oxpath.org

JCDL'17, Toronto, Ontario, Canada

DOI: 10.475/123_4

to dynamically load additional content into the current web page or to navigate to an entirely new one. Line 1 renders a seed web source. Line 2 identifies volume elements with the XPath selector `volume_path` and extracts relevant data with the extraction marker as in `:<volume>`. Line 3 interacts with the current volume container with the action step `{action}` and acquires the nested issue-related information with the use of a predicate, i.e., the construct `[]`. By analogy, line 4 progresses to the next level of journal components and additionally contains a Kleene star (`()*`). This construct enables the iteration over a paginated list of articles, e.g., by selecting and clicking a button to access the next page. Lines 5-6 extract articles with their respective data fields.

The wrapper evaluation outputs a result tree, which in our case has three main layers (except the root): 1) volumes, 2) issues, and 3) articles with corresponding attributes such as publication year, issue number, title, authors, pages, etc. This output tree can be converted into a representation appropriate for further processing.

3 OXPATH IN THE TRENCHES OF DBLP

With the dblp crawling architecture, the OXPath template above cannot always be applied directly for harvesting journal metadata. Not all content provided by publishers is automatically eligible for extraction. New publications have to be identified before their metadata are crawled and referenced in dblp. Such intercepting filters suggest the decomposition of the original OXPath template. For instance, the content platforms for journal publishing usually present their lists of issues (LOIs) with links to the individual tables of contents (TOCs). These URLs allow the generic expression to be decomposed in two expressions.

Starting from the page with a list of all issues (`URL_TO_LOI`) for a given journal, the first expression extracts the URL for the individual tables of contents and accompanying metadata:

```
1 doc('URL_TO_LOI')
2   /volume_path:<volume>
3   [./{action}/issue_path:<issue>
4     [./link_path:<url>:<metadata>] ]
```

The collected information, e.g., the volume and issue numbers, is standardized by the dblp crawling architecture. Only newly released issues are selected from the extracted URLs. The second template is used to evaluate an OXPath expression by specifying `URL_TO_TOC` for each URL in the filtered set. Thus, the extraction process is resumed on the level of articles with the table of contents on each visited page.

```
1 doc('URL_TO_TOC')
2   /(paginator_path/{action})*
3     article_path:<article>
4     [./{action}/data_path:<data>]
```

By using OXPath, the maintenance of these extraction steps is simplified significantly. With traditional dblp wrappers, both regular expressions and wrapper-specific code become invalid if publishers decide to use a new content platform or structurally change their websites. With OXPath these changes usually require adapting the XPath selector to the new representation of volumes, issues, articles, and accompanying data fields in the DOM tree. Action steps can be added and adapted in the expression templates analogously if user interaction is required. The following OXPath expressions specify the templates presented above for journal metadata extraction from the Cambridge Core[4] platform and illustrate the intuitiveness of this adaptation process.

```
1 doc('https://www.cambridge.org/core/...')
2   //*[contains(@href, 'panel')]:<volume>
3     [./following-sibling::*//a:<issue>
4       [.:<url=qualify-url(@href)>
5         :<metadata=string(.)>] ]
```

```
1 doc('https://www.cambridge.org/core/journals/...')
2   /(//hr[1]//following::*[@class='pagination'][1]
      //a[contains(., 'Next')]/{clkwithchange /})*
3     //div[@data-prod-id]:<article>
4       [.//li[@class="title"]/a[@class="part-link"]
          :<title=normalize-space(.)>]
5       [? .//li[@class="author"]:<authors=
          normalize-space(string-join(./a
          [@class="more-by-this-author"], ', '))>]
```

The extent of variety among publisher websites requires a set of variations of the generic expressions above. More characteristic expressions[5] illustrate this divergence and the advantages of OXPath. This demonstration will show how OXPath expressions for arbitrary publisher web sites are incrementally constructed and applied to extract metadata for dblp.

4 CONCLUSION AND FUTURE WORK

OXPath is a suitable tool for creating robust wrapper components in the face of the huge variety of interactive publisher websites for bibliographic databases such as dblp. Due to their declarative nature in the spirit of XPath, OXPath expressions are convenient for selecting the frequently occurring nested web page elements prototypical of the bibliographic domain. Parameterized templates can be specified with suitable XPath selectors for these typical repetitive structures. If page layouts are changed, these selectors can be re-specified in the templates accordingly. Action steps, in turn, can be added to simulate user interaction as necessary.

Wrapper maintenance tasks can be facilitated further by implementing tools assisting catalogers in devising OXPath expressions from templates. Furthermore, the overall approach can be improved by automating expression decomposition for links, e.g., leading to the pages of individual articles to extract additional data fields. These partial expressions then can be executed repeatedly to retry data extraction from error-prone detail pages, or evaluated in a distributed way to increase efficiency.

ACKNOWLEDGMENTS

This work was supported by the DFG grant LE 1088/1-2 and the EPSRC programme grant EP/M025268/1.

REFERENCES

[1] Tim Furche, Georg Gottlob, Giovanni Grasso, Christian Schallhart, and Andrew Jon Sellers. 2013. OXPath: A language for scalable data extraction, automation, and crawling on the deep web. *VLDB J.* 22, 1 (2013), 47–72. DOI: http://dx.doi.org/10.1007/s00778-012-0286-6

[4] https://www.cambridge.org/core

[5] See also http://dblp.uni-trier.de/papers/oxpath-examples-dblp.pdf for an accompanying paper with more examples of OXPath expressions.