BibSLEIGH: Bibliography of Software Language Engineering in Generated Hypertext

Vadim Zaytsev
vadim@grammarware.net

Universiteit van Amsterdam, The Netherlands

Abstract

The body of research contributions is vast and full of papers. Projects like DBLP help us navigate through it and relate authors to papers and papers to venues. Relating papers to papers is already considerably harder, and projects like Google Scholar do their best to battle all the variations in citations. Relating papers to topics is an open problem with some automated methods under development but mostly manual contributions. Relating papers to concepts and especially concepts to concepts is impossible without expert intervention and sometimes requires years of research to accomplish in a convincing manner.

BibSLEIGH has started in 2014 as a personal project for pretty-printing, normalising and eventually annotating bibliographic items. It quickly started displaying the properties typical for model repositories, both good and bad, linked to data extraction, consistency management, synchronisation, analysis, etc. At SATToSE 2015 I would like to show the project's current state and discuss its potential in a broader scope.

Motivation

After noticing how much time that could have been spent writing papers, is wasted on reformatting the bibliography, polishing it towards consistency, choosing the right abbreviations and making similar last minute decorative adjustments, I have started a project called BibSLEIGH for collecting and managing my bibTEX database. The main reasons were as follows:

- ◇ It is too late to start using advanced software like Mendeley: properly tagging several thousands papers and books is an impossibly massive investment.
- ◇ Obtaining individual bibTEX files (downloading from publishers' websites, copying from DBLP, searching in your old papers, crafting manually) takes time and provides no overview.
- ◇ BibTEX providers commit to questionable conventions: DBLP separates proceedings from inproceedings; ACM abbreviates venues; Elsevier includes abstracts; IEEE adds keywords; etc.
- ◇ Venue names have too many variations, including obviously unacceptable ones with the status (proceedings, post-proceedings, pre-proceedings, revised papers, special issue, etc) at the end or lacking, but location and dating information included in excruciating details.
- ◇ Different formats are needed in various circumstances: internal reports may have longer and more precise items; in-community links are contracted first to save space, etc.

In: A. H. Bagge (ed.): Proceedings of SATToSE 2015, Mons, Belgium, 6–8 July 2015, to be published at http://ceur-ws.org

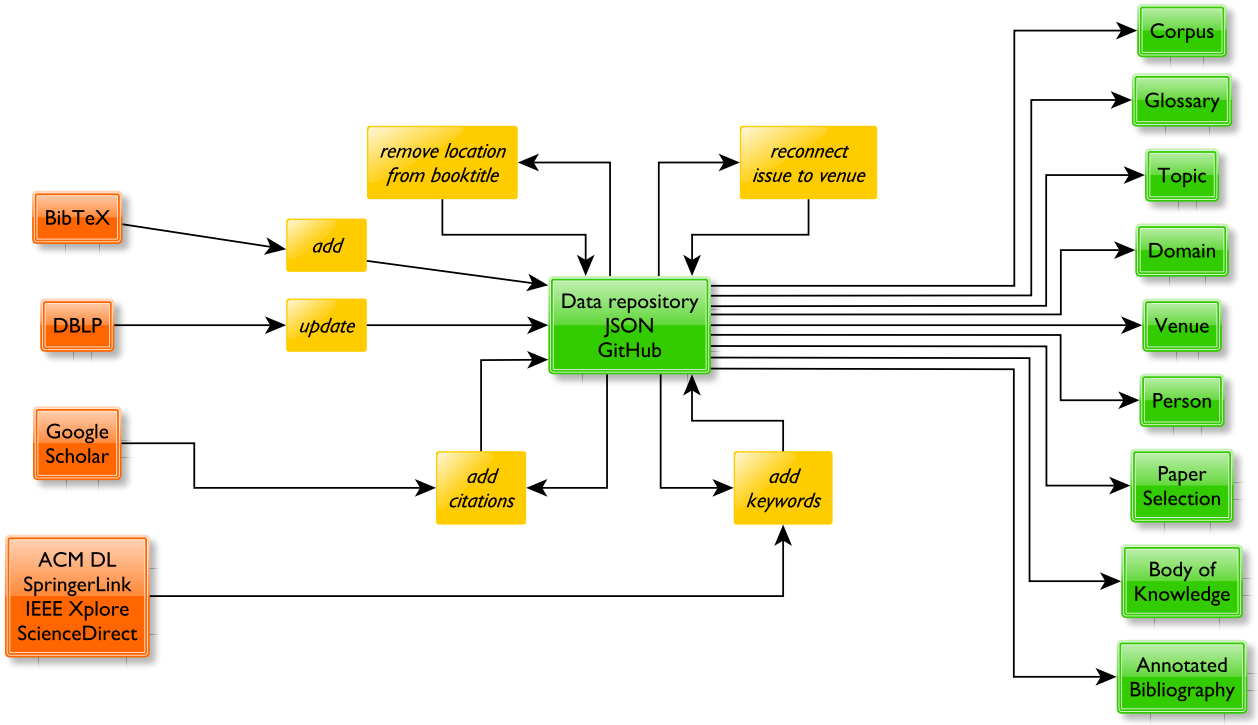

Figure 1: *The workflow of BibSLEIGH with concrete examples. On the left, in red, data sources: DBLP pages and its XML dump, bare bones noncontextualised BibTeX entries, Google Scholar-like hubs and publishers' websites. In the centre, the main BibSLEIGH repository in the heart of all related activities. On the right, in green, possible uses of the database if we assume it exists and allows for easy access, querying, bundling, annotating and representing. The yellow (in monochrome, light frameless slanted text) are examples of mappers that import content from various sources, improve existing content based on heuristics or enhance it with additional data.*

Solution

A lightweight solution was infeasible: most bibTEX providers resist crawling. At the same time, their data dumps are gigabyte long XML documents ruling out any free XPath processor. However, an iterative event-based parser was used to extract and sanitise interesting fragments — the initial case study covers 20 venues out of 6500+ available at DBLP with 25000+ papers. It was revealed that some of the irregularities are too sudden and impossible to detect, so the chosen solution is semi-automatic and produces a persistent intermediate representation which is stored separately and can be adjusted in any desirable way. A hypertext frontend is generated statically from this corpus of collected bibliographical data: it provides overviews, generates the actual bibTEX with desired conventions and links back to the editable corpus. Some examples of its activities are given below.

Incremental updates are possible with tools like `dblp2json.py` that slowly crawl a DBLP page of an event, collect all XML bibliographic files referenced from it, tries to figure out which one is the main entry (the issue of a journal or a volume of proceedings) and which ones are its elements, produces JSON enties accordingly and connects them to the rest of BibSLEIGH. Other such importers exist, and more can be written.

Automated transformation. When a venue which is already covered by the corpus, gains a new edition of a conference, it can be imported automatically. Subsequent sanitation might be necessary to polish some details, but the initial porting is a matter of a simple filtering and mapping of structured data. The main difference between this and the previous item is the necessity of having synchronising traits — i.e., matching previously imported content even if it evolved and import only new entries, — to make the project viable in the long run.

Semi-automated transformations. Suppose we found the title "Software Language Engineering - 7th International Conference, SLE 2014, Västerås, Sweden, September 15-16, 2014. Proceedings" unappealing and change it to a much more appropriate "Proceedings of the Seventh International Conference on Software Language

Engineering" (the year and the abbreviation is available from other fields, and the exact geographical location is irrelevant after the conference took place and tickers were reimbursed). This change itself is quick if done manually but its propagation from the proceedings entry to all the papers it contains, can be automated. There are many other situations when we need to intentionally disrupt and then automatically restore the consistency of BibSLEIGH.

Heuristic-based transformation. Page information is partially missing from some DBLP data: only the starting page of each paper is available. However, if we know the starting page of the next paper of the same volume, the ending page is easy to determine — and the heuristic is very realistic since modern CS/SE publications rarely contain advertisements between papers and rarely combine several papers on one page.

section gives an overview of several activities and artefacts that are or can be involved in the workflow of BibSLEIGH. Many boxes on the figure are mere examples, the system can be extended considerably.

Future work

What makes BibSLEIGH become more than a glorified wrapper for DBLP is harvesting its domain specificity and community specificity. While keeping the automated, semi-automated and heuristic-based transformations as maintenance activities, we can continue ingraining the bibliographic entities and their groups with information relating them to one another, as well as to concepts, methods, frameworks, approaches and toolkits. Annotating them manually or automatically with information about topics can aid automated clustering and linking beyond traditional methods depending on the citation information. We see this as another step towards the construction of a body of knowledge for the domain of software language engineering (SLEBoK).

Software language engineering, besides being a subdomain of software engineering, is known to be a bridging area of research, where a fair share of activities is devoted to seeking similarities between technologies and technical spaces, and to developing techniques with wide and cross-space applicability. However, even within one space reaching a point of soundly relating concepts can take substantial time and effort — consider laying relations between attribute grammars and affix grammars [10] or between object algebras to attribute grammars [12]. Can BibSLEIGH facilitate this?

At the same time, providing interactive access to the curated annotated corpus of academic papers on programming language theory, compiler construction, metaprogramming, software evolution and analytics, refactoring and other related topics can serve as an entrance point into the research domain as well as the foundation for some metaresearch activities. Its relevance and usefulness for the SLE community will be vastly determined by the direction the project will take, but in general all these questions are currently counted among the open problems of SLE [2]. The main questions the project is currently facing is whether to proceed with it beyond bibTEX pretty-printing and curation and if yes, then in which direction. Which features, if any, would be in demand within our (SATToSE+) community?

Related work

There exists related work in at least four different directions: model repositories; community support; annotated bibliographies; taxonomies. Model repositories such as FMI (Free Model Initiative) [13], ReMoDD (Repository for Model Driven Development) [8], CDO (Connected Data Objects)[1], Atlantic Metamodel Zoo[2], Grammar Zoo [16], GenMyModel[3], re on a quest of collecting models for various purposes. We can find quite a number of initiatives related to *community* management and facilitation: DBLP [11], Reengineering wiki [6], Researchr [14], Research 2.0 [1], SL(E)BOK, etc, including ones specifically aimed at community analysis, such as metaScience[4]. They usually combine requirements elicitation with experience reports with calls to arms. Annotated *bibliographies* also exist on different topics, like domain specific languages [5], reverse engineering [3], metrics [15], software (language) engineering[5], parsing [9], some of them covering up to 1700 papers! Such bibliographies and are usually constructed by senior experts and can require years of research to accomplish in a convincing manner. *Taxonomies* go even one step further and tie key publications together with key concepts and relations between them: examples exist for taxonomies of domain specific aspect languages [7], reverse engineering [4], (un)parsing [17].

[1] https://eclipse.org/cdo/
[2] http://www.emn.fr/z-info/atlanmod/index.php/Zoos
[3] https://repository.genmymodel.com
[4] http://atlanmod.github.io/metaScience/
[5] https://github.com/slebok/yabib

References

[1] D. Avrilionis, G. Booch, J. Favre, and H. A. Müller. Software Engineering 2.0 & Research 2.0. In P. Martin, A. W. Kark, and D. A. Stewart, editors, *Proceedings of the conference of the Centre for Advanced Studies on Collaborative Research (CASCON)*, pages 353–355. ACM, 2009.

[2] A. H. Bagge and V. Zaytsev. Open and Original Problems in Software Language Engineering 2015 Workshop Report. *SIGSOFT Software Engineering Notes*, 40:32–37, May 2015.

[3] M. G. J. v. Brand, P. Klint, and C. Verhoef. Reverse Engineering and System Renovation. *SIGSOFT Software Engineering Notes*, 22(1):57–68, Jan. 1997.

[4] E. J. Chikofsky and J. H. Cross II. Reverse Engineering and Design Recovery: A Taxonomy. *IEEE Software*, 7(1):13–17, 1990.

[5] A. v. Deursen, P. Klint, and J. Visser. Domain-Specific Languages: An Annotated Bibliography. *SIGPLAN Notices*, 35(6):26–36, 2000.

[6] A. v. Deursen and E. Visser. The Reengineering Wiki. In *Proceedings of the Sixth European Conference on Software Maintenance and Reengineering (CSMR)*, pages 217–220. IEEE Computer Society, 2002.

[7] J. Fabry, T. Dinkelaker, J. Noyé, and E. Tanter. A Taxonomy of Domain-Specific Aspect Languages. *ACM Computing Surveys*, 47(3):40:1–40:44, Feb. 2015.

[8] R. France, J. Bieman, S. Mandalaparty, B. Cheng, and A. Jensen. Repository for Model Driven Development (ReMoDD). In *International Conference on Software Engineering (ICSE)*, pages 1471–1472, June 2012.

[9] D. Grune and C. J. H. Jacobs. Complete Literature References and Additional Material to "Parsing Techniques: A Practical Guide — Second Edition", 2011. http://dickgrune.com/Books/PTAPG_2nd_Edition/Additional.html.

[10] C. H. A. Koster. Affix Grammars for Programming Languages. In H. Alblas and B. Melichar, editors, *Attribute Grammars, Applications and Systems*, volume 545 of *LNCS*, pages 358–373. Springer, 1991.

[11] M. Ley. The DBLP Computer Science Bibliography: Evolution, Research Issues, Perspectives. In A. H. F. Laender and A. L. Oliveira, editors, *Proceedings of the 9th International Symposium on String Processing and Information Retrieval, 9th International Symposium*, volume 2476 of *LNCS*, pages 1–10. Springer, 2002.

[12] T. Rendel, J. I. Brachthäuser, and K. Ostermann. From Object Algebras to Attribute Grammars. In *OOPSLA*, pages 377–395. ACM, 2014.

[13] H. Störrle, R. Hebig, and A. Knapp. An Index for Software Engineering Models. In S. Sauer and M. Wimmer, editors, *Poster Session of MoDELS 2014*, volume 1258 of *CEUR Workshop Proceedings*, pages 36–40. CEUR-WS.org, 2014.

[14] E. Visser, S. Vermolen, and E. van Chastelet. Researchr, 2009. http://researchr.org.

[15] R. Whitty. Object-oriented Metrics: An Annotated Bibliography. *SIGPLAN Notices*, 31(4):45–75, Apr. 1996.

[16] V. Zaytsev. Grammar Zoo: A Corpus of Experimental Grammarware. *Fifth Special issue on Experimental Software and Toolkits of Science of Computer Programming (SCP EST5)*, 98:28–51, Feb. 2015.

[17] V. Zaytsev and A. H. Bagge. Parsing in a Broad Sense. In *Proceedings of MoDELS*, volume 8767 of *LNCS*, pages 50–67. Springer, Oct. 2014.

Resources

- http://bibtex.github.io — web front end
- http://github.com/slebok/bibsleigh — partially curated JSON data
- http://github.com/bibtex/bibsleigh — JSON refactorings and visualisations