



ulm university universität
uulm

Ulm University | 89069 Ulm | Germany

Faculty of Engineering,
Computer Science and Psychology
Institute of Media Informatics
Visual Computing Group

Combining Interactive Exploration and Search for Navigating Academic Citation Data

Master's Thesis in Computer Science at Ulm University

Presented by:

David '-1' Schmid

david-1.schmid@uni-ulm.de

<https://orcid.org/0000-0001-8453-5026>

Examiners:

Prof. Dr. Timo Ropinski

Prof. Dr. Birte Glimm

Advisors:

Christian van Onzenoodt

Dominik Meißner

2018/19

Last updated May 27, 2019

© 2018/19 David '-1' Schmid

This work is licensed under the Creative Commons **Attribution 4.0 International** License. To view a copy of this license, visit <http://creativecommons.org/licenses/by/4.0/>.

Typesetting: XeLaTeX; I use Arch, BTW.

Abstract

In this thesis, we build an academic search engine which supports exploratory research by focussing on tracking connections through a wider variety of aspects. These aspects are different than what is commonly available through for-profit providers of academic search engines. Especially when researching unfamiliar territory, we need to identify key journals, authors and papers without losing track of our collected sources and their context. Commercial providers of academic search engines calculate their own heuristics and metrics to recommend other publications or authors for further study, but these might not satisfy a researchers curiosity. On order to find different views through other aspects of the data, we will need to build our own database. To achieve this, we collect freely licensed data from several sources and serve them publicly accessible through our web application. We build software, that can summarize metadata and structure research automatically, by visualizing the taken path of research steps as a graph. Through it, we can retrace out steps and preserve the context of newly discovered sources. The publicly accessible system is built by using and building free and open software, so it is available for everyone to use and extend.

Contents

Contents	v
I Introduction	I
1.1 Goal	I
Functional Requirements	2
Non-Functional Requirements	2
2 Interactive Navigation in Practice	3
2.1 Starting from a Clean Slate	4
2.2 Detail View	4
Publication Elements	4
Facet Elements	5
Emitters	6
2.3 Navigation Graph	6
2.4 Advanced Query Methods	7
2.5 Conclusion	8
3 Data Sources	9
3.1 Microsoft Academic Graph (mag)	9
3.2 dblp computer science bibliography (dblp)	10
3.3 Semantic Scholar (s2)	10
3.4 Notable Mentions	11
3.5 Conclusion	12
4 Prototype	13
4.1 Front-End	13
Vue.js	14
Cytoscape.js	14
4.2 Back-End	15
Apache Solr	16
4.3 Middleware	19
Django Channels	19
5 Related Work	21
5.1 Reference Management	21
5.2 Exploration	22
6 Evaluation	25
6.1 Method	25

Basic Tasks	25
Advanced Tasks	26
6.2 Results	26
Participant Remarks	27
6.3 Discussion	27
7 Future Work	29
7.1 Metrics	29
7.2 Load Balancing	29
7.3 User Interface Improvements	30
7.4 Staying Updated	30
8 Conclusion	33
List of Figures	35
List of Tables	35
Bibliography	37

One

Introduction

“go away ? tell all others not in the list below to stay out!”

robots.txt,
sciencedirect.com

When breaking into unfamiliar fields of study, researchers need to survey the current state-of-the-art, identify important authors and find influential journals. To achieve this, researchers commonly rely on services offered by for-profit corporations that have the resources to crawl large portions of the web, including scientific publications. Their academic search engines offer a variety of additional functionality to aid the exploration process. Besides search, they allow seeing how often a publication was cited and by whom, enabling us to find related articles. Based off of their collected data, some additionally offer recommendations for related papers, topics or keywords found through text analysis.

But sometimes, what they offer in functionality is just not enough to satisfy our curiosity. Google Scholar, for example, has two options for sorting: by relevance or date. If we opt to sort the result by date, we will only be presented with results that were submitted within the last year, with no option to show more. Where Google fall short, Microsoft fill the gap: Microsoft Academic provides much more options for sorting and filtering. When showing the results for a query, they give an overview of authors and journals, associated topics and keywords, as well as the options to apply these as filters. Then again, if researchers wanted or needed to have more options for filtering they are left to their own devices. For example: when viewing a list of citations for a paper, none of the search engines allow to *exclude* a specific author from them (i. e. to remove self-citations from the results).

Exploratory research also includes reading many publications and deciding their relevance for the own research topic. To keep track of the initially large amount of publications, researchers currently lack tools that integrate tightly with search engines. To our knowledge, there is no software that automatically tracks the relations between publications, so researchers have to use other tools like mind maps or spread sheets for conducting structured surveys.

1.1 GOAL

We propose to combine major aspects of structured research into one system: search, exploration and reference management. Integrating these into one system, will improve on the current shortcomings of academic

search engines and reference management solutions. Each of these major aspects have excellent tools that support us in accomplishing these tasks in isolation, but they lack the powers of the other two. While there are tools that try to reach into one of the other domains, their support is not integrated as a defining feature, but a crutch to moderate tedious work.

To disambiguate search, exploration and reference management further: we define *searching* as the process of using known words as a query to find matching publications. *Exploration* is everything that happens afterwards: following references for further perusal, learning new vocabulary for common techniques and identifying important authors or journals. What was learned through exploration might in turn feed back into new searches; tracking this process is what we refer to as *reference management*.

Functional Requirements

versatile filtering

We want to provide more *versatile filtering*, so we have better control over our results. Another goal is enabling users to keep track of their research and exploration in academic data through automated means. A search engine that integrates *automated tracking* of our movement while researching, enables us to retrace our steps. Identifying influential conferences and authors is an important aspect of exploring other fields of study. To empower researchers to decide these criteria on their own, we integrate *alternative metrics* into the search engine.

automated tracking

alternative metrics

Non-Functional Requirements

open

For-profit search engines are usually limited in their filtering capabilities that are available through an API; they might not even have an API. In essence, they are hindering access to several facets of the data, which might be useful to researchers. To meet the ends, we need convenient and programmatic access to large data sets.

intuitive

Powerful search capabilities usually entail complex query languages. These capabilities and languages easily overwhelm users that are unfamiliar with both, the language and the storage back end. Consequently, it is crucial to provide meaningful building blocks that abstract from the query language, while avoiding unnecessary restrictions.

maintainable

Once finished, we want to provide the open source implementation including instructions to replicate the setup. To encourage others to further improve the infrastructure and add modifications, software maintainability is a major contributing aspect. Every dependency of the system should have an active community, such that software rot can be countered as a collective.

expandable

In order to allow the system to grow beyond its initial prototype, the implementations must allow adding redundancy and load-balancing facilities. The mayor involved components must be able to distribute load for accepting client requests, serving the front-end and handling internal coordination.

Two

Interactive Navigation in Practice

“You may not [...] automatically search and index the [...] meta data [...].”

**Terms of Use,
SpringerLink**

The navigation front-end allows researchers to build a graph of their exploration process: each node represents a taken step, with edges tracing their taken path. This inherently structures the research process and provides insights on how information was discovered. The graph reflects previous actions and visualizes them accordingly.

The navigation graph can be constructed from six primary node types (Figure 2.1), which will trigger the display of details about the associated metadata. These details are then used to inspect the information and make decisions to take further steps, thus expanding the graph. When reviewing a publication’s references (the active node’s data), a researcher might find a contribution relevant to their interest. Once clicked, a new paper node (📄) is placed with an edge connecting to the publication it was discovered through, thus documenting the exploration automatically.

- 📄 paper
- 📍 venue
- 📖 journal
- 🔑 keyword
- 🔍 search
- 👤 author

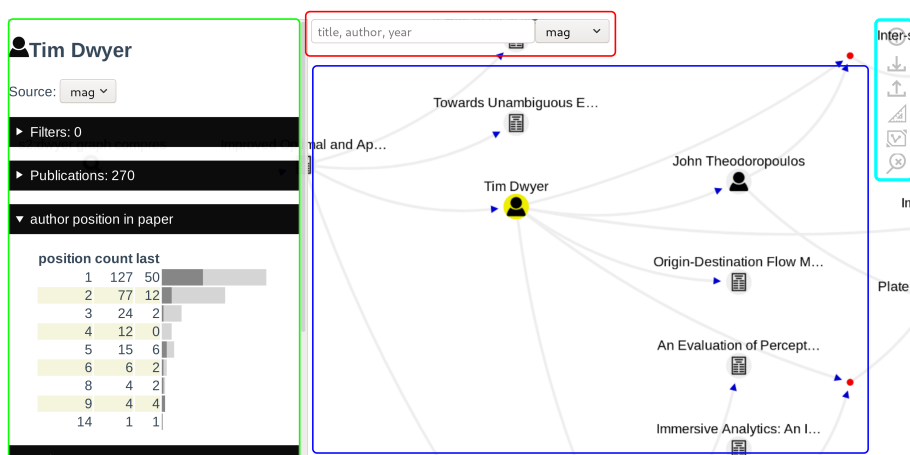


Figure 2.2: Screenshot of the implemented prototype

the search input with collection selector (red, top center), the tools (cyan, upper right), the navigation graph (blue, two right thirds of the image) and detail view (green, left third)

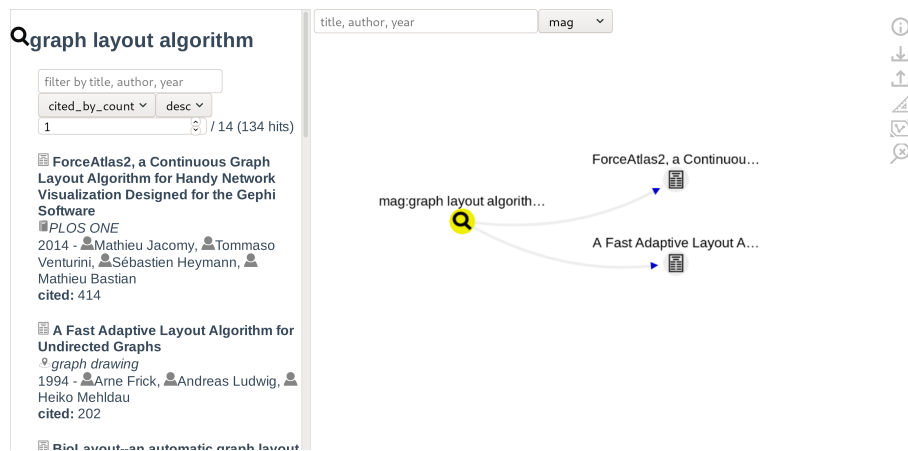


Figure 2.3: The UI after some basic initial interaction.

This is the result of searching for *graph layout algorithm* and clicking two papers, linking them to the search node. The active search node (Q) is highlighted with yellow background.

2.1 STARTING FROM A CLEAN SLATE

Upon first visit, users are presented with only the search box and the tools (Figure 2.2). The only possible node type that can be added to the graph is a search node (Q), since the other five primary node types can only be accessed through the detail views. To place a new search node (Q), researchers enter their terms into the search box.

The node is automatically marked as selected, which triggers the display of the side-pane containing the search results. From this side-pane, researchers may select relevant papers by clicking on the titles, causing them to be added to the graph as nodes (P), with edges pointing from the search. Figure 2.3 displays the search results for *graph layout algorithm*, where a researcher selected two additional papers from the results: they are linked to the search node (Q).

2.2 DETAIL VIEW

A detail view will always consist of two elements: its type as an icon, followed by its name. While the data displayed in a detail view differs by the selected node type, they always have one or more elements that concerning associated *publications*.

Publication Elements

A researcher's goal is to find relevant publications, hence each node will list these and provide mechanisms to filter them further. An author node (A) will list all publications, where the person's name is listed as an author. Journal nodes (J) contain a list of all papers published, the same holds for venue nodes (V). A keyword node (K) will list all publications that were associated with the keyword. For a paper node (P), we provide up to two elements for publication data: one for publications that were *referenced by* the selected paper and one for publications that the selected paper was

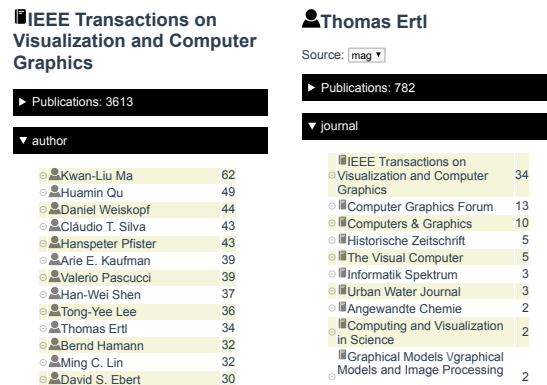


Figure 2.5: Facet views on journal and author nodes

Left: to which journals (and how often) has an author published. **Right:** which authors published in the journal (and how often).

cited by (Figure 2.2). If information about references or citations is not available, the elements are not displayed.

These publication lists are accompanied by elements that manipulate the contents of the list. Researchers have different criteria by which they select publications: some prefer seeing recent publications first, others would rather see the most cited papers. To achieve this, researchers may sort the list either by title, citation count or year, and change the sorting direction to their preference.

Additionally, these lists can be filtered further by using the filter text-input. For example, one might be interested in a publication that is referenced by the currently inspected paper. To record the connection between these papers in the graph, a researcher can use the filter box to find the referenced paper by entering fragments of title, author and year.

Facet Elements

Another type of element that is provided through the detail views are *facet* elements: these can request more information about publications that are associated with a node, and are usually concerned with counting occurrences of specified attributes. For exploratory research, one might find it interesting to see where an author usually submits their writing, or in which years this author had published. The faceting elements can summarize the publications of a node by several aspects: *author*, *year*, *journal*, *venue* and *keywords*. Unfolding *author* in the detail view of a journal node (📁), will show a list off all authors (👤) that ever published there, sorted by how often they did. The reverse holds as well: unfolding *journal* on an author node (👤), will show all journals (📁) this author published to, including the count. Both views are displayed in Figure 2.5.

Author nodes (👤) have another special element that will summarize their position within the list of all authors. This was implemented, because some fields of study give special value to how often an author named is first or last on a publication. For that reason, we provide a summary about the positions where the author occurs within their publications (Figure 2.6).

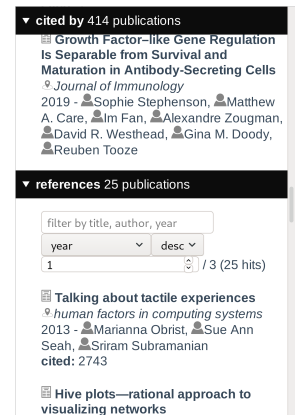


Figure 2.4: Two publication elements on a paper's detail view.

The upper element lists the publications that the currently selected paper was cited by, the lower lists which sources the paper references.

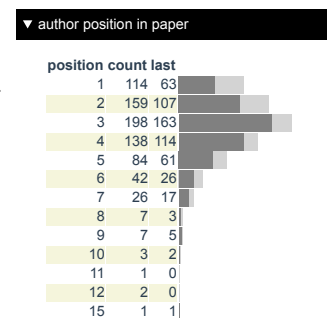


Figure 2.6: The summary of an author's position within their publications.

Here, the author has 114 publications where they are in first position, 159 in second, etc. The column *last* counts the instances where the author was mentioned last in the list of authors.

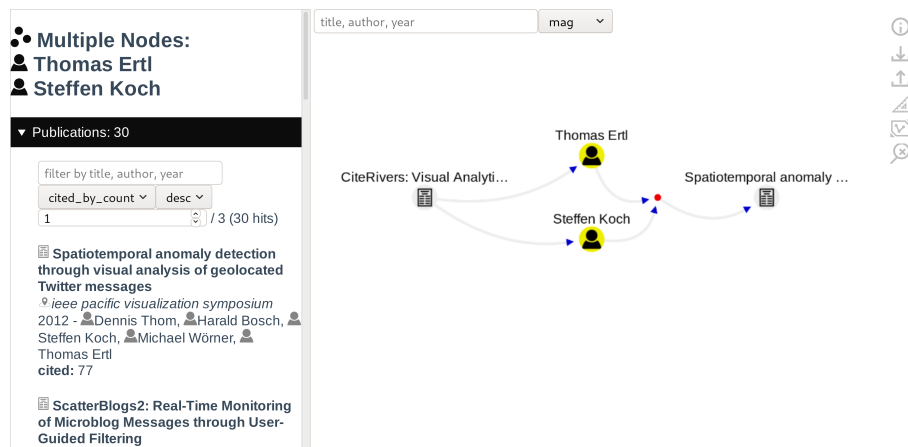


Figure 2.7: The detail view when multiple nodes are selected

When multiple nodes are selected, the search result is the intersection of the results for each respective node. The small red node connects multiple parents to nodes that were placed from such a detail view.

Emitters

Emitters are the most integral part of the detail views: these are the elements which cause new linked nodes to appear. They are identified by having an icon in front of their name, signaling which kind of node they will emit onto the graph. Navigating the available metadata is the core of this thesis, these are the elements that build the graph and enable the exploration. When one of the detail views contain some information regarding the author *Thomas Ertl*, this is represented as Thomas Ertl and when the mouse cursor is placed above the emitter the element will change to Thomas Ertl, hinting its function.

Emitters are found everywhere within the detail views to enable researchers to follow their interests based on facets or by publications. Each major node type can be found within the detail views, except search emitters (Q): they are exclusive to the search box, and are always the beginning of a journey.

2.3 NAVIGATION GRAPH

The navigation graph can be panned and zoomed as desired, and is fully visible as long as no node is selected. Once a node is selected, the previously explained detail view will appear. The circular context menu can be used to delete unused nodes; it will be shown when a node is clicked with the right mouse button. For touch screens with no right click, touching and holding a node will also show the menu.

Researchers can select several nodes to request details about multiple entities in conjunction. For example: when a paper's co-authors () were placed on the navigation graph, selecting two or more author nodes at the same time will request all publications where the selected authors collaborated. Holding down a **Ctrl** or **Shift** key before clicking on a node will add or remove them from a selection. This is not limited to author nodes, the

The figure displays three panels, each showing search results for 'cited by 173 publications' with different filters applied. Each panel includes a search bar, a dropdown menu for sorting, and a list of results with their titles, journals, authors, and citation counts.

- Panel 1 (Left):** Filter: `+(author:"Manfred Reichert")`. Results include:
 - Process time patterns** (Information Systems, 2016) by Andreas Lanz, Manfred Reichert, Barbara Weber (cited: 15)
 - PQL - A Descriptive Language for Querying, Abstracting and Changing Process Models** (International Conference on Enterprise, Business-Process and Information Systems Modeling, 2015) by Klaus Kammerer, Jens Kolb, Manfred Reichert (cited: 4)
 - Change and Compliance in Collaborative Processes** (Ieee international conference on services computing, 2015) by Walid Fdhila, Stefanie Rinderle-Ma, David Knuplesch, Manfred Reichert (cited: 14)
- Panel 2 (Middle):** Filter: `-(author:"Manfred Reichert")`. Results include:
 - A comparative study of workflow customization strategies: Quality implications for multi-tenant SaaS** (Journal of Systems and Software, 2018) by Majidi Makki, Dimitri Van Landuyt, Bert Lagaisse, Wouter Joosen
 - Coadapting multidimension process properties.** (Journal of Software: Evolution and Process, 2017) by Djamel Eddine Khelladi, Reda Bendraou, Regina Hebig, Marie-Pierre Gervais
 - Modeling Contextualized Flexible Cloud Workflow Services: An MDE based approach** (research challenges in information science, 2017) by Yosra Lassoued, Selmin Nurcan
- Panel 3 (Right):** Filter: `year:[2016 TO *]`. Results include:
 - Process Instance Similarity: Potentials, Metrics, Applications** (OTM Confederated International Conferences "On the Move to Meaningful Internet Systems", 2016) by Johannes Pflug, Stefanie Rinderle-Ma (cited: 3)
 - Integrating Business Process Management to Model Context in Healthcare: A case study using periperative processes** (2016) by Amos Harris
 - Automatic Business Process Test Case Selection: Coverage Metrics, Algorithms, and Performance Optimizations** (International Journal of Cooperative Information Systems, 2016) by Kristof Böhmer, Stefanie Rinderle-Ma

Figure 2.8: Examples of advanced queries

Left to right: `+(author:name)` to show only self citations; `-(author:name)` to show all, except self citations; `year:[2016 TO *]` to show publications from 2016 onwards

selection can also include other node types: Selecting an author (👤) and a journal (📖) will show all publications that this author contributed to the journal.

When an emitter from a selection of nodes is invoked, all currently selected nodes are connected to the new node via a *multi* node (•) (Figure 2.7). This will keep the clutter in check and allows to select all parents at once: when such a multi node (•) is selected, it will select all of its parents.

2.4 ADVANCED QUERY METHODS

Searching and filtering are amplified by advanced queries. Through them, researchers can filter publication lists according to their needs. For example: when a paper node (📖) is selected, researchers might show interest about the citing publications (the *cited by* element). To remove all self citations, the researchers can insert `-(author:"Capitalized Name")` into the filter text input. When desired, the `-` can be substituted by `+`, to show all self-citations. Range queries that limit the publications to selected years are supported as well, `year:[2016 TO *]` will only show publications from 2016 onward (Figure 2.8).

The system also supports queries that contain only the first few characters of a word. This is important, when different spellings of a word might occur, such as *visualisation* and *visualization*. When searching, a request “*visuali*” will match both variants and others that begin with this fragment (i. e. *visualizing*).

Last, but not least, we can use the *Levenshtein distance* to find variations in words. Because, as much as we try: there will always be a typo around that no one found. By using this distance we can include these misspelled

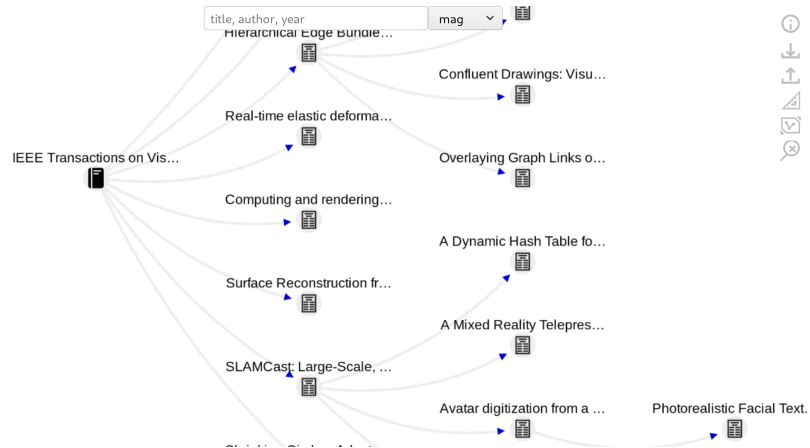


Figure 2.10: A possible result during breadth-first research

Starting from a journal, several papers published in the journal were selected for further inspection. Once read, the relevant references were put on the graph, extending the exploration tree in depth.

Q tessellation~1 -tessellation

Source: mag

Filters

filter by title, author, year
year desc
1 / 21 (206 hits)

- Geometric and combinatorial structure of a class of spherical folding tessellations - II.
Australasian J. Combinatorics
 2018 - Catarina P. Avelino, Altino F. Santos
- Identification and temporal tracking of droplet clusters based on Voronoi tessellation and mean shift algorithm
Proceedings 18th International Symposium on Flow Visualization
 2018 - H. Lian, Y. Hardalupas, Xy. Chang
- Strong Isoperimetric Inequality for Tessellating Quantum Graphs
arXiv: Metric Geometry
 2018 - Noema Nicolussi


Figure 2.9: Query example for finding all publications that misspelled *tessellation*

`tessellation~1` will find all matches with edit distance equal or smaller than one and `-tessellation` will remove all correctly spelled matches.

results. Adding the suffix `~1` or `~2` to a search term will include words with an edit distance *lower or equal to* one or two, respectively (Figure 2.9).

2.5 CONCLUSION

Our application combines powerful filtering techniques with a graph visualizing relations between the taken actions. It is useful for identifying relevant publications, their authors and publishing institutions. It will also support in structuring and organizing for a breadth-first search into a given topic (Figure 2.10).

While the application automatically stores the graph in the browser's local storage, researchers may use the upload button  in the toolbox to store the graph on our servers, so they may access it from another location. The application is publicly accessible on <https://sonne.ods.de> and the source code is provided on github <https://github.com/sonne-academic>, licensed under the terms of the *Apache License 2.0*.

Three

Data Sources

“Institutional subscribers are NOT permitted to [...] Use robots or intelligent agents to access, search and/or systematically download [...]”

**Digital Library Terms of Use,
IEEE Xplore**

At the time of writing, there are several data sets that are available for free download. This chapter highlights the differences in both licensing terms and the provided data, an overview is provided in [Table 3.1](#). While there are excellent API providers that allow retrieval of publication metadata, we focus on data that is provided as a whole. They all share common attributes: the data about an author’s ORCID is mostly not present and they are usually distributed as lists of documents, rather than being normalized for relational databases. This might be owed to how the data is collected, since crawling the web will always yield a document. Most distributions do not include the abstracts as plain text, as they are subject to the same copyright restrictions as full-texts of research papers.

3.1 MICROSOFT ACADEMIC GRAPH (MAG)

The Microsoft Academic Graph data is freely licensed under ODC-BY 1.0 and receives regular updates (1 or 2 weeks between updates). It is the largest, freely available database and is distributed through Microsofts’ Azure Storage. Microsoft Academic will provide it to anyone free of charge, although their terms allow them to change that at any time. But, they will charge for storage, traffic and computation done on Azure.

The MAG is built through Microsofts’ bing search engine crawlers and are post-processed with artificial intelligence machine reader [\[Res19\]](#). Abstracts are provided as inverted index, which would allow reconstruction

	size	updates	format	citation	abstracts	license
MAG	210	1-2 weeks	tsv	✓	✓	ODC-BY 1.0
DBLP	6	nightly	xml	✗	✗	ODC-BY 1.0
S2	45	irregular	jsonl	✓	✓	custom
UPW	23	irregular	jsonl	✗	✗	none
CORE	123	irregular	jsonl	✗	✗	none
OAG	?	irregular	jsonl	✓V1 ✗V2	✗	ODC-BY 1.0
CSX	?	?	?	✓	✗	CC-BY-NC-SA 3.0

Table 3.1: Overview of the reviewed data sets.
Sizes are approximate and stated in millions of documents

of the original text. This allows them to provide the data under this free license, otherwise they would infringe on the copyright of the authors.

Drawbacks

Contrary to other data sources, their data is normalized for relational databases and is split into several tables. This complicates the import considerably, since our database requires them as a denormalized format.

Another drawback is the initial difficulty of requesting access to the data, since Microsoft Academic does not allow direct download of the set. Instead, one has to register for an account at their Azure Cloud, which only works with a Microsoft account. Although Microsoft do provide step-by-step instructions, the approximately 50 steps for set-up can be confusing. Once all the requirements are met, MAG data will be sent to a storage account of the user. This step requires sending an e-mail to Microsoft Academic, with the request for access.

Since we were not registered with Azure, we were granted 170 € of “free trial credit”, and did not have to pay for the cost. The accumulated cost was about 50 € for retrieving the uncompressed data (about 500 GiB) and the storage cost. Since this is (labor) expensive, we decided to utilize the permissive license to host it somewhere, where others have easier access (see [Aca19]).

3.2 DBLP COMPUTER SCIENCE BIBLIOGRAPHY (DBLP)

The data from dblp is also available under ODC-BY 1.0 and receives regular updates (nightly updates + monthly releases). It is much smaller than other sources, since the data is curated by dblp, and covers almost no events outside of computer science. dblp also has no data about references between publications, but their data is procured from reliable sources such as the publishers and sometimes manually submitted by volunteers [tea19]. Their XML is a list of documents, so it was straightforward to import.

Drawbacks

The XML-Schema that dblp uses is very simple, but the data sometime holds unfortunate surprises. To get these out of the way, our current process is to transform the source XML to jsonl format, with many assertions to catch problems before they are indexed.

Another minor inconvenience is that the XML is encoded as ASCII with UTF-8 replacements specified through the DTD. Conversion to UTF-8 was important, since all other data were in UTF-8 as well.

3.3 SEMANTIC SCHOLAR (S2)

The Open Research Corpus provided by Semantic Scholar¹ contains approximately 45 million publications, including references and abstracts. It is provided under a non-commercial custom license that allows use and the creation of derivative works for internal operation. Since they include the abstracts as plain text, they can no longer license the data as freely

¹<https://api.semanticscholar.org/corpus/>

(such as ODC-BY). When we contacted them to ask if our use was covered by the license, they responded that it is. Hence, we do not provide access to the abstracts, but they could be used to derive a more sophisticated recommendation for other papers.

The data is provided as jsonl format, and was the easiest to index. They also provide keywords that were derived automatically from the data. As sources, they list a variety of publishers and other freely available data sets; among them are dblp, mag and aminer [Sch19].

Drawbacks

Apart from the unusual license, updates to the data are provided at irregular intervals: the latest two versions (at the time writing) are dated 2019-01-31 and 2018-05-03. The keywords include seemingly random guesses such as “*Naruto Shippuden: Clash of Ninja Revolution 3*” which is assigned to more than 36.000 publications, and is mostly associated with journals that are concerned with potatoes (Figure 3.1).

There were minor quirks when importing the data, since the unique author ids were sometimes nested lists instead of lists. Since we did not want to rely on pre-generated ids, we discarded them and thus had no further problems.

3.4 NOTABLE MENTIONS

CiteSeerX (csx)

CiteSeerXs’ data is provided under *CC-BY-NC-SA 3.0 Unported*² terms. To access the data via their Google Drive, one has to reach out to them through their “Contact Us” form, which did not work. After several fruitless attempts to contact the various jointly responsible persons for csx, we tried contacting one of the collaborators that they were publishing with. They graciously provided us with the data, under the condition that we would not share it, so we did not want to use it.

The data is sourced by their own crawlers, which also generate references between publications. We are uncertain if csx is still actively maintained, since the github repository still receives minor updates. But when we issued a search for papers from 2019, we only received 17 results and some of them were actually from 1999.

CORE (core)

The CORE data³ is not published under any licensing terms, but accompanied by a disclaimer that “[...] it is up to the user of this dataset to ensure that the way in which they use the dataset does not breach copyright”. To our understanding, copyright claims for databases are still under dispute with not enough cases and landmark court decisions. Therefore we assert, that the collator of such a database holds the full rights to both distribution and derivative works. Thus, we did not attempt to use it or gather more information about it.

²<http://csxstatic.ist.psu.edu/downloads/data>

³<https://core.ac.uk/services/dataset/>

🔑 Naruto Shippuden: Clash of Ninja Revolution 3

▶ Publications: 36182
▼ Journal
○ American Potato Journal 942
○ American Journal of Potato Research 821
○ Acta Neurochirurgica 390
○ CoRR 288
○ Economic Botany 238
○ Acta diabetologia latina 208
○ Research in Computing Science 193
○ Brittonia 186
○ La Rivista Italiana della Medicina di Laboratorio - Italian Journal of Laboratory Medicine 183
○ World Journal of Surgery 174
○ Acta Endoscopica 141

Figure 3.1: Semantic Scholar tagged more than 36.000 publications with “Naruto Shippuden: Clash of Ninja Revolution 3”

Unpaywall (upw)

Unpaywall provide a service that allows checking if there are Open Access documents for a given DOI. They also provide snapshots⁴ of their data at irregular intervals. As with CORE, they issue no licensing terms with the data. When we contacted them about the terms they responded: “*We do not have an explicit license specified for the dataset, but you are welcome to use it any (legal) way you like!*”.

Aminer (am)

Aminer have several data sets for various purposes (not limited to academic citation metadata) and a comparatively small set (3 million papers) from 2017 derived from DBLP⁵. Their licensing terms are unclear, as no license is stated for most of their pages except one: the Open Academic Graph.

Open Academic Graph (oag)

The Open Academic Graph is a collaboration between MAG and AM, that also have links between the sets. There are currently two versions available:

v1 (2017-06)⁶ has references, but the information originally (before February 2019) was licensed as: “It’s a free product for the research community only”, with a remark on another page that it would be licensed under ODC-BY 1.0. This has since been fixed, but the original terms were not acceptable, since they were too vague.

v2 (2019-01)⁷ is not licensed as of 2019-05-16 and the reference data was removed from the part that was contributed by mag. Their Google Group contains a message that it is licensed under ODC-BY 1.0⁸, but that is not assuring enough.

3.5 CONCLUSION

DBLP, S2 and MAG were selected to be a part of our database, since they were correctly licensed. They have their own merits and drawbacks: DBLP is updated on a daily basis and has high standards towards quality of the metadata, but it’s limited to computer science publications and has no citation information. MAG is time-consuming to retrieve and difficult to index, but it provides the largest amount of papers and regular updates. S2 takes the middle ground, since it’s easy to retrieve and index, but not as large as the data from mag and does not get updates as frequently. Once the other publishers’ licensing is cleared up, we may expand the available sources further.

⁴<http://unpaywall.org/products/snapshot>

⁵<https://aminer.org/citation>

⁶<https://aminer.org/open-academic-graph>

⁷<https://www.aminer.org/oag2019>

⁸<https://groups.google.com/d/msg/open-academic-graph/p70XtIDVdMc/1qQwGPWYAgAJ>

Four

Prototype

“Using scripts or spiders to [...] harvest metadata [...] is a serious violation [...] and will result in the temporary or permanent termination of download rights for the subscribing institution.”

**Authorized Uses for Institutional Subscribers,
ACM Digital Library**

This chapter gives rationale about the architectural choices for each of the involved components. To provide a chance for others to reproduce the implementation and installation, we focus on free and open source software. This way, all components fulfill the non-functional requirement *open* (from [section 1.1](#)). The infrastructure is built as three tiers: storage and indexing *back-end*, the *middleware* for load distribution and back-end access, and a *front-end* for application logic and communication (as introduced in [chapter 2](#)).

4.1 FRONT-END

The front-end is written as a web-application, in order to achieve widespread access: web-browsers are common to all desktop environments and operating systems. Additionally, modern web-browsers provide a feature-rich run-time environment and eliminate the need to write and distribute a purpose-built desktop application. Apart from the built-in asynchronous event handling, they provide means for: external communication through WebSockets, persistent storage for user-data and configuration, rendering a large variety of different media formats, text rendering, and hardware-accelerated graphics.

They also come with disadvantages: the predominant standards HTTP, HTML, CSS, SVG, WebGL, ECMAScript and the browser API are very extensive. Additionally, some of these standards regularly receive updates, including breaking changes and arguably misguided decisions by their implementing vendors¹. This requires knowledge of all these standards and their functional interaction (i. e. CORS, CSP). Unfortunately, web-browsers tend to implement various sub-sets of these standards (depending on the browser version), further complicating development. To reduce this complexity of supporting various browser versions, the front-end supports the recent versions of both, Mozilla Firefox and Chromium. Both browsers are available under open licenses, for a large variety of operating systems, and support many of the modern versions of aforementioned standards.

¹<https://github.com/GoogleChromeLabs/airhorn/pull/37>

The web-application is built using two free and open frameworks: Cytoscape.js for the navigation graph and Vue.js for everything else.

```
<template>
  <details
    ref="details"
    @toggle="toggled">
    <summary>
      <slot name="summary">
      </slot>
    </summary>
    <slot
      v-if="open || alwaysLoad"
      name="detail"></slot>
    </details>
</template>

<script lang="ts">
import Vue from 'vue';
export default Vue.extend({
  name: 'SidebarDetail',
  props: {
    alwaysLoad: {
      type: Boolean,
      default: false,
    },
  },
  data: () => ({
    open: false,
  }),
  methods: {
    toggled(ev: any) {
      // ...
    },
  },
  computed: {
    details() {
      // ...
    },
  },
});
</script>

<style scoped>
summary {
  background-color: black;
  color: white;
  padding: 0.5em;
  position: sticky;
  top: 0;
  margin: 1em 0;
}
</style>
```

Figure 4.1: Vue.js combines markup, code and style into single files

Vue.js

When we started, we did not want to use a JavaScript framework at all, since every added dependency increases the cost for maintenance. Instead, we wanted to build mostly on *Web Components*² and *Custom Elements*³.

Web Components solve performance issues in web-browsers: inserting an element into the Document-Object-Model (DOM) would trigger all update mechanisms. These updates can cause the logic of the web-application to recalculate the size another DOM element, causing further updates to the DOM. This is especially problematic when interacting with and visualizing data, as we rely on changing many elements in short intervals. Web Components work solve this, by introducing a *shadow DOM*. A shadow DOM is detached from the document's DOM, providing a mostly inert container. Child elements of a shadow DOM are not affected by style or events of the DOM it is contained in, and vice-versa cannot affect the real DOM. Custom Elements additionally allow the definition of new HTML tags (i. e. `<fancy-container>`) with their own predefined style and behavior, effectively providing a mechanism for scoping style and reusing code. Regrettably, they currently cannot be used when a web-server does not allow *unsafe inline* evaluation, as they require it for the style application.

We chose *Vue.js*⁴ over other JavaScript frameworks, because its usage is very close to Web Components and can be replaced with plain *HTML Modules*⁵ once they are decided by the W3C. It enables to combine style, code and markup into *single file components* (see [Figure 4.1](#)), closely mimicking the planned feature set of HTML Modules.

Cytoscape.js

With the introduction of the `<canvas>` element, browsers provide us with a single element that can render bitmaps without triggering DOM updates. But, the canvas does complicate development of interactive applications, since we are no longer working with a tree of HTML elements. This prevents us from using many of the convenient features that HTML, CSS and JS provide in combination: styling and event handling has to be implemented, yet again.

*Cytoscape.js*⁶ is a graph theory library for visualization and analysis [[Fra+19](#)]. It offers built-in animation framework, a styling framework similar to CSS, event handling and outstanding quality of documentation. Since it is built to work on a canvas element for efficiency, these features are important. Where browser support is available, Cytoscape.js will also spawn an extra thread for drawing on the canvas. In case the main rendering loop of the browser is under heavy load and cannot provide a stable frame-rate,

²https://developer.mozilla.org/en-US/docs/Web/Web_Components

³<https://developers.google.com/web/fundamentals/web-components/customelements>

⁴<https://vuejs.org/>

⁵<https://github.com/w3c/webcomponents/blob/gh-pages/proposals/html-modules-explainer.md>

⁶<https://js.cytoscape.org/>

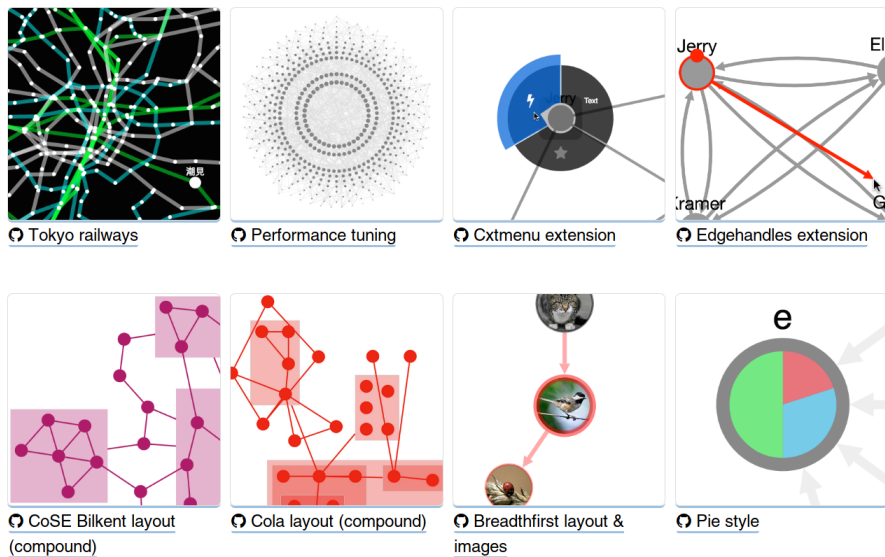


Figure 4.2: A selection of Cytoscape.js extensions

the additional thread will use the *offscreen canvas* API, to keep the interaction with the graph smooth. Due to its pluggable architecture, Cytoscape.js supports additional graph layout algorithms and extensions (e. g. context menus, see Figure 4.2). While Cytoscape.js does provide efficient built-in layout algorithms, it does not provide a layout algorithm that will produce stable results: everytime the graph changes, the nodes might be scattered randomly. Thus, we use the *cytoscape.js-dagre*⁷ extension for a stable hierarchical layout of the navigation graph.

4.2 BACK-END

In order to serve the data to our front-end, we need to use a database capable of storage and fast access to 210 million publications. Due to the circumstance, that most sets are distributed as lists of documents we favor databases built for storing documents. To allow the system to grow and provide reliable service, the database must be able to distribute its storage onto several hosts.

There are several popular candidates for document storage, indexing and search: *Apache Solr*⁸, *Elasticsearch*⁹, *Sphinx*¹⁰ and *Xapian*¹¹. These four were selected from a larger set of Information Retrieval systems, as their features aligned most with our goals. Sphinx was discarded early, since it does not allow replication of the index. While Xapian allows replication, they disqualified for their lack for distributed search and computation.

⁷<https://github.com/cytoscape/cytoscape.js-dagre>

⁸<https://lucene.apache.org/solr>

⁹<https://www.elastic.co/>

¹⁰<https://sphinxsearch.com/>

¹¹<https://xapian.org/>

Elasticsearch and Apache Solr are both based off of the same indexing back-end, written in Java, and support distributed storage, retrieval and replication. Both are supported by an active community (*maintainable*) and allow indexing a wide variety of document formats, including PDF and Open Document Files. They also provide web APIs and client libraries for several programming languages. The Solr framework is built with a focus on providing a pluggable architecture (*expandable*). As Elasticsearch requires a commercial license for clustering the database onto multiple hosts, we decided to use Apache Solr.

Apache Solr

Solr splits its databases into collections of documents. During usual operation, each collection is defined by a schema which represents the fields of a document and their types. A most basic example just defining an id and its type:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema name="default-config" version="1.6">
  <fieldType name="string" class="solr.StrField" docValues="true" sortMissingLast="true"/>
  <uniqueKey>id</uniqueKey>
  <field name="id" type="string" indexed="true" required="true" stored="true"/>
</schema>
```

The `fieldType` definitions can be much more complex to support and optimize for different use cases. To implement the prefix matching for our front-end, we use the type `text_prefix` with two different analyzers: for query, we split the string value along its whitespaces and convert it to lower-case to provide case-insensitive results. The generated tokens are then used for searching the index for matches. The `index` analyzer is used when we submit the document to Solr. Instead of storing the lower-cased words, we add an additional filter that will generate *n-grams* from the left word boundary.

```
<fieldType name="text_prefix" class="solr.TextField" multiValued="false">
  <analyzer type="query">
    <tokenizer class="solr.WhitespaceTokenizerFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
  <analyzer type="index">
    <tokenizer class="solr.WhitespaceTokenizerFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.EdgeNGramFilterFactory" minGramSize="2" maxGramSize="15"/>
  </analyzer>
</fieldType>
```

These *n-grams* are used to provide partial matches of a word, as demonstrated in [Figure 4.3](#). Solr's documentation explains examples and facilities for complex use cases and different languages: its example configuration¹² features common configurations for various languages, and multiple analyzers for different matching solutions in English (i. e. phonetic similarity).

¹²<https://github.com/apache/lucene-solr/blob/master/solr/example/example-DIH/solr/solr/conf/managed-schema>

Figure 4.3: The index analyzer with an EdgeNGram filter will produce several values that are indexed and can be used to match partial words. Here “Visualization” is split into several values, and querying for “visualiz” will match one of the indexed tokens (highlighted in orange).

Once the matches are retrieved, Solr will assign a score to the matches. The score depends on the amount of matched tokens (by default, a search result contains all documents that match at least one) and their position in relative to other matches. We influence this score by *boosting*, favoring full matches of a word over partial matches (n-gram) matches. During query, we define a score boost function as “ngram_field word_field^10” to boost the score of a fully matched word by an order of magnitude, and thus show it earlier in our search results.

boosting

Solr’s *faceting* is the mechanism through which our web-application retrieves the counted attributes for its detail views. Given a query for an author (author:“Timo Ropinski”), we can specify additional fields for requesting facet counts. When we ask for counts for the year field, Solr will run the query for the author and count how often each respective value for year occurs. This aids both, fast filtering of results and the motivating exploration aspect. Facets can be requested for any attribute of the documents, including (but not limited to) publication year, associated keywords, journals or venues of the publications; essentially any field that was stored in the index. But, faceting comes with a cost: it requires to store the individual attributes besides the inverted index, thus increasing the necessary drive space. Currently, our index of MAG, S2 and DBLP consumes roughly 380 GiB of storage (95 GiB per host). Such an index can triple in size during indexing or an optimization pass, but these are the compacted sizes.

faceting

The large amount of data is difficult to survey before it is stored: incomplete or misunderstood documentation about the structure of the data can make it difficult to import it into a database in the first place. To counter this, Solr offers a *schemaless* operation, that can infer previously undefined fields and guess their type. This was of great benefit to our first attempts, since we were able to peek into the data and the occurring edge cases. We used schemaless mode to gain initial insights into the data, to then define an optimized schema by observation rather than guesswork.

schemaless

Why not Graph or Relational Databases?

The decision to forego graph and relational databases for the back-end was deliberate: relations between nodes are many (i. e. autor → (publications, coauthors, journals), journal → publications), and (depending on the data set) require a large amount of pre-processing. This is due to the fact, that most sets are distributed as non-normalized documents (as discussed in [chapter 3, Data Sources](#)).

Another problem with these connections is especially tricky to solve: should authors sharing the same name be treated as one? If not, which decisions must be taken to safely distinguish the authors from their name-sakes'? When we inspected the data, we found several spelling variations as well as mistakes in authors' names, owed to both, publishers and authors. As seen in [Figure 4.4](#), *Klaus H. Hinrichs* exist in variants with and without their middle name. Two publications of *Frank Steinicke* were erroneously recorded with misspelled names and distributed with wrong metadata by the publishers. By choosing a document database we can effectively defer these inconsistencies, to address them while exploring the data; whereas relational databases force us to address these problems before indexing, in order to allow their optimization features to work.

*Titan*¹³ (former JanusGraph) approaches this shortcoming of graph databases by adding Apache Lucene based indexers (Solr and Elastic Search) to its supported storage back-ends (Apache Cassandra, Apache HBase or Oracles' BerkeleyDB). While combining the best of both worlds, it comes at the price of sharply increased complexity. Using Titan would enable running graph data algorithms like PageRank, in a distributed platform (Apaches' Spark, Giraph and Hadoop), enabling faster computation of h-indices or journal impact factors. Although desirable, the complexity was weighed against the added value and deemed too *difficult to maintain*, replicate and implement in the duration of a master's thesis.

Hardware Recommendations

The system started out with a search cluster of three Solr instances (8 GB RAM, 200 GB persistent storage and 4 vCores, each). It is enough to provide reliable and fast access to a medium-sized publication database (46 million documents, s2).

Unfortunately, the introduction of the largest available data set (210 million documents, MAG) was too much for these three nodes. They were frequently running out of RAM for the same basic queries that ran smoothly for the medium-sized data set. Upgrading the cluster to four bigger nodes (16 GB RAM, 250 GB persistent storage and 8 vCores, each), improved the situation considerably. With that change, we still consider the system feasible to deploy for researchers and other enthusiasts with access to 64 GB of RAM.

General recommendations on sizing the hardware are difficult to pin down exactly, since there are many variables involved [[Luc12](#)]. The Elasticsearch documentation¹⁴ recommends keeping the RAM per node below 32 GiB, because Java will use *Compressed Ordinary Object Pointers* to

```
"facet_fields": { "author": [
  "Timo Ropinski",37,
  "Klaus H. Hinrichs",36,
  "Frank Steinicke",35,
  "Gerd Bruder",9,
  "Jennis Meyer-Spradow",4,
  "Jörg-Stefan Praßni",2,
  "Frank Steinicke",1,
  "Frank Stenicke",1,
  "G. Brudei",1,
  "Harald Frenz",1,
  "Jörg Mensmann",1,
  "Klaus Hinrichs",1,
```

Figure 4.4: Misspellings and variations in authors' names

Klaus H. Hinrichs have been recorded in one instance where their middle name was not listed and *Frank Steinicke* have two recorded typos that were not corrected by the publishers.

¹³<http://titan.thinkarelius.com/>

¹⁴<https://www.elastic.co/guide/en/elasticsearch/reference/7.0/heap-size.html>

reduce memory consumption per allocated object. When the node stays below 26GiB Java will use the even more efficient *Zero-Based Compressed Ordinary Object Pointers*¹⁵. Once above the threshold, the JVM will need to use an extraordinary amount of additional RAM just for storing 64-bit pointers instead of 32-bit pointers plus offset [Lan14]. They also recommend to only use half of the available memory for the JVM heap, so the kernel can use the remaining memory for the filesystem cache. Beside the above, Solr and Elasticsearch are very reserved on giving actual recommendations beyond trial and error.

4.3 MIDDLEWARE

The middleware serves two purposes: It's foremost function is to serve the WebSockets established with the front-end and abstract from Solr's API. In the not too distant past, Solr has had several security vulnerabilities that allow denial-of-service and remote code execution. The servers must be protected from unauthorized access, which is the secondary function of the middleware.

The middleware is implemented in Python, as modern Python provides native asynchronous programming models and the run-time environment is available for a majority of operating systems. Additionally, Python is easily extended by the large amount of available libraries and frameworks. The web-framework *django*¹⁶ provides many management features necessary for the middleware (such as session management, object-relational mapping and basic security measures). It is complimented by the *Channels*¹⁷ and *aiohhttp*¹⁸ frameworks: they extend django with asynchronous WebSockets and asynchronous HTTP, respectively.

Django Channels

To fully utilize the django extension, we need to provide a *channel layer*. This enables Channels to hand the message distribution off to a message broker for load balancing, publish-subscribe messaging and at-most-once message delivery. Additionally, these layers allow the addition of background tasks, i. e. spawning a worker process that will send updates to Solr. Channels does not include such a layer in its distribution, although they maintain an implementation for *Redis*¹⁹. There is also a community managed layer extension for *RabbitMQ*²⁰. Such a layer is not strictly necessary for Channels to function, but it will ease scaling the system when necessary. Redis was not used as a layer, since this would require yet another service (*maintenance*), the RabbitMQ layer was not available when we started the project, so we implemented an in-memory layer for *ZeroMQ*²¹. This layer can be switched from the currently set `inproc://memory` address to an address of an external host, in case the system needs to grow.

¹⁵<https://docs.oracle.com/javase/10/vm/java-hotspot-virtual-machine-performance-enhancements.htm>

¹⁶<https://www.djangoproject.com/>

¹⁷<https://channels.readthedocs.io/>

¹⁸<https://docs.aiohttp.org/>

¹⁹https://github.com/django/channels_redis

²⁰https://github.com/CJWorkbench/channels_rabbitmq

²¹<http://zguide.zeromq.org/>

WebSockets

Communication between web application and Channels is implemented similar to *JSON-RPC*²² over WebSockets. The choice for using WebSockets is motivated by two reasons: firstly, due to its smaller footprint on traffic when compared to HTTP/1.1. Secondly, to enable the server sending messages that were not necessarily generated by a client request. This is important if we were to make use of Solr's streaming feature which can generate arbitrarily large responses. To minimize RAM usage on the web-server, we would split the JSON stream into several responses, removing the necessity of storing the whole response on the server. We can also use these WebSockets, if we decide to expand the system to allow collaborative, simultaneous editing of a navigation graph instance.

²²<https://www.jsonrpc.org/specification>

Five

Related Work

“arXiv [...] provides metadata for all submissions which is updated each night shortly after new submissions are announced.”

Open Archives Initiative (OAI),
arxiv.org

In this chapter, we focus on tools for reference management and techniques for visualizing and exploring academic citation data.

5.1 REFERENCE MANAGEMENT

We focus on free and open source software that is actively developed and available for all major operating systems. Additionally, we require that these solutions provide means to interface with external sources: this way, we might be able to integrate them into our system. With these rules in place, we are left with three solutions: JabRef, Zotero and BibSonomy.

JabRef is a reference manager written in Java that stores its database as BibTeX or BibLaTeX file, but is able to import and export to various other formats for reference management. It features integration with a variety of word processors and allows interfacing with several external search engines such as Google Scholar, dblp, pubmed, Crossref and *more*¹. The search results can be automatically imported into the reference management software. Furthermore, it can acquire related articles through *Mr. DLib*, an open-source project that provides recommendations as a free service. It also supports reading and changing PDF annotations for attached files, as well as XMP Metadata. Their Firefox plugin for sending citation data to JabRef currently only supports windows. Since they store all information as BibTeX files, they do not support indexing PDF content.

*Zotero*² is a reference manager that allows to store its database offline, as well as synchronize it with online storage. It will index the content of attached PDFs to allow free text search through the content. Their software is complimented by connector plugin for web-browsers, relying on the Zotero application running in the background. The plugin supports a variety of publishing sites, and can import the PDFs directly when the user is allowed access through a subscription. When the website is not known to a converter, Zotero also allows to create a citation to the web page, including a snapshot of the site. Besides its reference manager, Zotero provides a cloud storage for synchronization. This can be used combined with the browser plugin to save references to their cloud, in the event the

¹<https://www.jabref.org/#features>

²<https://zotero.org>

Zotero application is not running in the background. They provide 300 MiB of storage for free, larger capacities can be acquired with an annual fee. Zotero has an open API that can be used to manipulate the data in the online storage. Collaboration in groups is possible and is synchronized promptly between the clients via WebSockets.

*BibSonomy*³ is an online service for social bookmarking and publication sharing by the University of Kassel. Their service allows importing publication data via a browser extension, uploading files (for private or group access only) and encourages public sharing of collected citation data. This way, users can search the public repository for already entered metadata and add them to their own repository. There is an *user* that mirrors the dblp repository, so some metadata is automatically populated. They have a collaboration feature, so open or closed groups of users can collect and annotate data jointly. Their online storage for privately stored documents seems to have no limitation in size.

Summary

Zotero and BibSonomy have many features in common, but differ in minor details. Zotero allows adding metadata between items to indicate a relation. Relations can be added to the citation data, or items lower in the hierarchy (such as notes), but there is no way to specify the kind of relation. BibSonomy has a more powerful tagging feature than Zotero: they allow users to build hierarchies of tags, such that child tags are automatically filed into parent tags. Instead of a more powerful tagging system, Zotero builds such hierarchies by using a folder structure, to the same effect as BibSonomy. Although BibSonomy is an online service, it can be synced with a JabRef extension so users are able to work offline.

None of the presented solutions have facilities for tracking the users research, motivating us to implement our system.

5.2 EXPLORATION

The biggest influence for this thesis is the work of Kahng et al. [Kah+16]. Their system serves as visual aid for building SQL queries, and enable users to execute complex operations (i. e. SQL pivots) interactively. They continued their research for an interface that provides means for navigating large graph databases [Pie+17]. Their approach of not presenting the whole graph, but instead providing the users with means to visit only the parts relevant to the users, seemed promising and was taken further by this thesis. Both implementations are not available as open source, and they also use databases focused relations, rather than documents.

ReviewerNet is built to aid editors of scientific journals in finding suitable candidates for peer-reviewing submissions based on their field of study and past collaborations [SGC19]. The demonstration is based on a small subset of the s2 corpus: 17.754 papers and their respective authors, from eight journals concerned with computer graphics. They send the complete database on initial connection. This transmits a large amount

³<https://www.bibsonomy.org/>

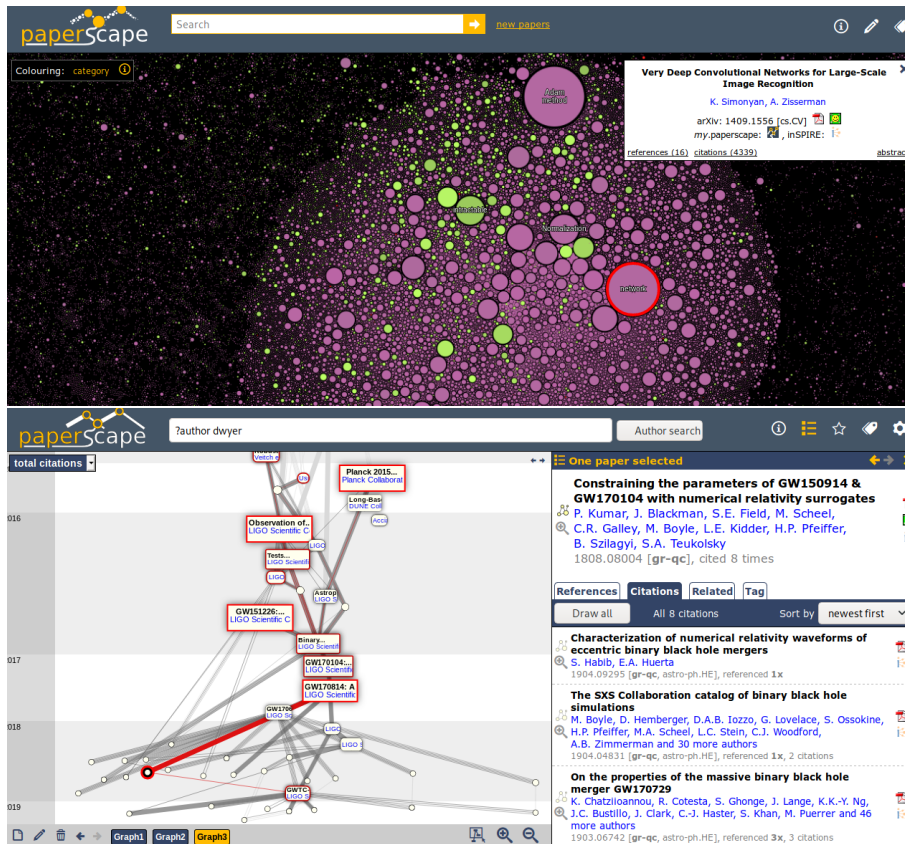


Figure 5.1: The two paperscape front-ends

Top: paperscape renders one circle per publication on arXiv. Sizes depend on how many times the paper was cited. **Bottom:** the “my paperscape” frontend traces connections between selected papers.

of data, but it keeps an editor’s choices and search requests private. While they chose to provide the source code for the system, they did not add a license to it. This prevents us from integrating ReviewerNet into our service, which would have given access to the whole database.

*Paperscape*⁴ is a tile based map of all publications on arXiv (Figure 5.1). Citation data is generated by extracting information from the PDF files. Publications are rendered as a circle, their respective sizes depend on how often they were cited, and the color depends on the field of study they were published in. The position of the circles is calculated offline through an n-body physics algorithm that places them closer to each other, when they are connected by a citation. There is also an accompanying front-end for exploration that visualizes citations between papers.

Juniper, a Multivariate Graph Visualization, uses a combination of trees and tables visualize connections between the nodes of a graph database [NSL18]. Their approach is very flexible and draws many connections between authors, thus allows to explore by many relationships. They built

⁴<https://paperscape.org> & <https://my.paperscape.org>

a graph database from a small subset of DBLP and enhanced it with additional attributes taken from <https://vispubdata.org>. Their front-end is very complex and requires its users either to read the paper or watch a video in order to use it. The freely licensed (BSD-3-Clause) *source code*⁵ is available, which allows reuse and modification. Regrettably, they include many dependencies and the original implementation is built on a graph library, so we chose to not try and to integrate it into our system.

The arXiv implemented a *bib overlay*⁶ for their website that automatically retrieves citation and reference data through Semantic Scholar (the source for the s2 data). It is deployed as an optional feature and to protect their users privacy, must be enabled explicitly.

The presented work of other researchers shows potential in many categories of exploration. But, none of them support tracking to the extent our prototype provides, which is why we built it.

⁵<https://github.com/caleydo/lineage/tree/juniper>

⁶<https://github.com/mattbierbaum/arxiv-bib-overlay>

Six

Evaluation

“We want to provide a public, open, and free API for all. [...] For that to work, we ask that you be polite [...].”

Etiquette,
Documentation for Crossref’s REST API

To evaluate the system, we invited seven computer science researchers to qualitative interviews with think-aloud testing. They were given tasks, which they were to solve using our system. In order to investigate if the system offers intuitive means of interaction, we made notes on how they accomplished these tasks; without giving an introduction to the system. Additionally, we explicitly mentioned that they are allowed to try anything else that they were curious about, or ask if it was possible.

6.1 METHOD

Participants were instructed to visit the prototype and were given a physical copy of a selected survey paper. To coax the participants into using the features of the system, we selected a paper with a long title and it many references (460) [BH18]. They were then asked to find this paper using the system and find out more details about the paper. Once the paper was selected and placed on the graph, they were instructed to find out more about the authors of the paper, and the journal it was published in. They were then asked to find answers to several questions. Once the tasks were completed, we asked if they wanted to share any remarks concerning the system.

Basic Tasks

Regarding one author of the publication:

- which other authors were they collaborating with?
- in which journal did they publish most?
- which publication of the author is most cited?
- in how many publications were they the primary author?

Regarding the journal:

- which is the most cited paper in the journal?
- in which year did the journal publish most?

- which author did publish most on this journal?

Once finished, participants were asked to select one of the referenced papers in the physical copy of the publication. They were then tasked to add this paper to the graph and add a note to the reference.

Advanced Tasks

After the basic tasks were finished, they were given additional tasks:

- Find out, how many papers were written in collaboration by the two authors of the survey, and add the most recent to the graph.
- Find an author of the most cited paper of any journal; how often did the author publish to this journal?
- Is there a year when this author published more than once to the journal?

6.2 RESULTS

The first three participants were given no introduction to what the prototype does, except that it can be used as an academic search engine and that the search supports matching by prefix (rather than having to type complete words).

After the first three participants were finished, it became clear that they did have a very hard time navigating. For one, when they were asked to find out something about the author, they clicked on the authors name; rather than the icon (at the time of the interview, only the icons were able to emit new nodes). When asked why they chose to use the text, consensus was due to the habit of using other search engines. When they were then asked to find out more about the journal, all had a hard time even finding the name of the journal.

The advanced tasks are tailored towards selecting several nodes at once, but without explanation were very difficult as well. Once they solved the first task by filtering the publications for the other authors name, they were asked: “what would you expect to happen, when you select two nodes?” The responses were: “you can do *that*? That would have been useful to know beforehand!”

Once told and without asking for instructions, they were immediately using the Ctrl key to select a second node, except one participant: they tried to move one node on top of another to achieve this, hoping that they would be grouped together.

There was another commonly encountered design flaw of the evaluated system: when participants clicked on the paper they were supposed to find, they were expecting to be presented with the details immediately. Instead, the system drew the new node on the graph, without any feedback except the animation. Due to the animation, every participant took note that something happened, but showed no intent of interacting with the new node. Without knowing, that the system was built to support tracking the path they took, the author and journal nodes were (in some cases) placed as children of the search node, instead of the associated paper.

After three completed interviews, we came to the conclusion that we should at least explain the intent of the system and which problems it tries to solve, *before* they were given their tasks. Once they completed the basic tasks, we asked what they would expect to happen if they were to select more than one node. This led to the same exclamation as before, all participants started experimenting by themselves, and began to answer some advanced tasks; even before they were told what to solve next.

Participant Remarks

All (except one person with no subjective need for reference management) were stating, that the system is useful for them, and they could see themselves using it in the future for overcoming various tasks. Several mentioned that this would be useful for doing a breath-first search for structured literature assessment, since they could easily track their path among the surveyed publications. Others mentioned that the system would be very helpful when writing grant proposals, since this often requires them to write about of the state-of-the-art in other fields of study. If they receive the grant, the system would help them to recover their tracks at a later time. Several participating researchers are using Zotero and were asking if they could import their already established references or export it to Zotero.

6.3 DISCUSSION

Once introduced to the more of the advanced features of the system, most participants saw a great deal of value for the prototype, especially for structured research. All participants were able to derive that holding Ctrl will change the mode to selecting multiple nodes by themselves. We suspect this is due to their familiarity in interacting with computers, since all participants had a background in computer science.

After the interview's results were collected, we changed the system interaction in two ways. For immediate feedback, we changed the emitter's behavior to display the details immediately. Additionally, we changed the emitters to place new nodes when the name of the attribute is clicked, instead of just allowing emission via its icon.

All participants completed the given tasks within 10 minutes or less, but most of them stayed for an equally long time to learn more about our system. They also suggested more improvements and eagerly requested features, such as the integration into existing reference management solutions.

Seven

Future Work

“Demonstrate an elevated level of commitment to open-data and open research [and] Become part of a global data-sharing community, learning, collaborating, and advocating with a leading-edge network of data research experts [...]”

Member Responsibilities,
DataCite

With the prototype and the backing infrastructure, we created a new platform for navigating and exploring data. The well of ideas did not run dry yet, there several further ideas that could be implemented.

7.1 METRICS

Exploring through the basic metrics in the already implemented prototype is already very useful, but during the evaluation and our own usage, we found more interesting metrics that help with exploration. Every time a list of publications is displayed, we could add another filtering mechanic: the faceting of Solr allows us to see which author or journals are prominent within a full text search, as well as the years when the results were published. This would help both, in selecting new filters to narrow down the result and the exploration aspect, since we can see a summary of the metadata. Especially for an often cited (> 1000) publication, it could be interesting to see which journals or authors the references came from.

When an author has publications under more than one name, either by accident or due to a name change, the author nodes should also carry the different aliases and calculate their metrics accordingly. A similar system could be in place for journals that are stored with the year in front of their name, but that could as well be implemented by using regular expressions instead of storing a list of all instances. Regular expressions would also have the benefit to catch future instances of the journal, instead of adding them manually to the list.

7.2 LOAD BALANCING

While the current implementation does have facilities for load balancing, we don't make use of them. The nginx web server has only one upstream source configured, and the ZeroMQ channel layer is configured to keep its messages in memory, rather than distributing it over several machines. Requests to Solr are only sent to a single node, which forwards the requests to the correct Solr shard. Using a balancing algorithm to distribute the

requests to all available nodes would take some load off of this instance, but there was no need to configure it, yet.

7.3 USER INTERFACE IMPROVEMENTS

When the explored graph grows larger, it would be helpful to collapse parts of the graph, to reduce visual clutter and aid summarization of an explored subgraph. To maintain a better overview, tagging and coloring nodes by category could be beneficial. Additionally, a searchable list of all explored nodes and assigned tags could be useful for quickly navigating between nodes. Visually grouping nodes in hierarchies could be an option as well, and could be automated by utilizing such tags.

As compression inherently reduces entropy for more efficient storage, its property can be leveraged to lessen cognitive load by reducing the *edge count* between nodes [Dwy+13]. This yields a cleaner appearance and provides visual clustering to the observer. The authors tested their approach successfully with small graph sizes (< 2000 nodes and edges), so this might find re-use for collapsing research paths. Another related approach reduces the *node count* of graphs through bundling them by their context [HWR18].

Search results could also be clustered by topic, which Solr can derive automatically for a given query. Since clustering is a CPU bound operation, it adds a few seconds of latency and should be provided as an optional action. Once this is implemented, balancing the load to all Solr nodes becomes more important.

Integration with external web services for reference management (such as Zotero or BibSonomy), would allow users to import their already researched materials, as well as adding new materials to their reference management of choice. Zoter's web services also implement versioning and notifications about updates, which can be used to keep the information consistent and the graph in sync. This is especially important, once the users are allowed to collaborate on our system.

Conveying the metadata of the selected node within the page's markup (e. g. Dublin Core) would enable browser plugins of reference managers to store the reference, without requiring the user to connect to an online service. This would support users that chose to use reference management that is not bound to an online service, such as JabRef.

As mistakes sometimes happen, a possibility to undo the previous operation would further improve the system. Navigation between previous and future states could be coupled to the browser history.

7.4 STAYING UPDATED

When the data is not kept relatively recent, the usefulness of the system degrades over time. Since the MAG is difficult to index into Solr and its retrieval is associated with fees from Azure, it might be beneficial to look for other alternatives or find a cheaper way to retrieve and index updates.

Crossrefs' EventData¹ could be such an alternative, as they allow bulk requests for new submissions or comments of various sources. They also

¹<https://www.eventdata.crossref.org/guide/>

collect Twitter messages and posts on reddit that mention DOIs or landing pages of a conference, as well as new publications registered with DataCite (a major DOI provider) that are not limited to publications, but also include published data sets. This data is very suitable for ingestion into Solr, as all information provided as stand-alone documents. Referencing information between documents are also available, iff the publishers chose to allow public access. The EventData service is relatively new and still in *beta* stage: as soon as it is stable, it will provide new opportunities for sourcing new publications.

If our system was popular and connecting an active community, we could allow researchers to submit corrections and missing references. Ideally, we could pass these on to the publishers, either through Crossref or DataCite. Alas, such functionality is prone to misuse and needs moderation, which brings us full circle: that would require an active community.

Eight

Conclusion

“Wow, that is awful!”

***On: how most academic publishers work,
Myself***

We present our novel exploration system for *public use*¹ and its freely licensed *source code*² to encourage adoption and modification. We hope that it might evolve, not just through our contributions and visions, but also through community effort. Even if the user facing portion is not what others had in mind, we provide a platform with access to metadata, so others may build their application based on it.

We achieved our goals for the prototype, to provide more *versatile filtering* and *alternative metrics* about the metadata. Furthermore, we met our requirement for *automated tracking* of the steps taken in the exploration of the data. Our non-functional requirements have been fulfilled: it is *open, maintainable* and built on an *expandable* foundation. The evaluation through qualitative interviews helped to address some shortcomings of the user interface and showed favorable regard by the participants.

While we now have a web service that serves access to this data, we hope that more publishers will opt to provide the data not just to selected search engine crawlers, but to the research community as well. But for now, we researchers either have to rely on for-profit corporations to give back to the community, or we demand that scientific publishers should provide metadata freely to the public. We opened the data of the Microsoft Academic Graph to a wider audience, by hosting it on *Zenodo*³. Since then, it has been retrieved more than 600 times, helping the research community.



In this thesis, we present an open source prototype for exploring academic publication information through different data sources. It is designed to permit both, extension in functionality and supporting infrastructure. We proposed options for keeping the data recent and gave recommendations for interfacing with external applications.

We plan to maintain and develop the system further, so it might grow and improve.

¹<https://sonne.0ds.de>

²<https://github.com/sonne-academic>

³<https://zenodo.org/record/2628216>

List of Figures

2.1	Icons and meaning of the primary node types that can be placed on the canvas.	3
2.2	Screenshot of the implemented prototype	3
2.3	The UI after some basic initial interaction.	4
2.5	Facet views on journal and author nodes	5
2.4	Two publication elements on an paper’s detail view.	5
2.6	The summary of an author’s position within their publications.	5
2.7	The detail view when multiple nodes are selected	6
2.8	Examples of advanced queries	7
2.10	A possible result during breadth-first research	8
2.9	Query example for finding all publications that misspelled <i>tes-sellation</i>	8
3.1	Semantic Scholar tagged more than 36.000 publications with “Naruto Shippuden: Clash of Ninja Revolution 3”	11
4.1	Vue.js combines markup, code and style into single files	14
4.2	A selection of Cytoscape.js extensions	15
4.3	The index analyzer with an EdgeNGram filter will produce several values that are indexed and can be used to match partial words. Here “Visualization” is split into several values, and querying for “visualiz” will match one of the indexed tokens (highlighted in orange).	17
4.4	Misspellings and variations in authors’ names	18
5.1	The two paperscape front-ends	23

List of Tables

3.1	Overview of the reviewed data sets.	9
-----	---	---

Bibliography

REFERENCES IN 3: MICROSOFT ACADEMIC GRAPH (MAG)

- [Aca19] Microsoft Academic. *Microsoft Academic Graph*. 2019-03-22. DOI: [10.5281/zenodo.2628216](https://doi.org/10.5281/zenodo.2628216) (cit. on p. 10).
- [Res19] Microsoft Research. *Microsoft Academic*. 2019. URL: <https://www.microsoft.com/en-us/research/project/academic/> (visited on 2019-05-16) (cit. on p. 9).

REFERENCES IN 3.1: DBLP COMPUTER SCIENCE BIBLIOGRAPHY (DBLP)

- [tea19] dblp team. *How accurate is the data in dblp?* 2019-09-15. URL: <https://dblp.org/faq/13500484.html> (visited on 2019-05-16) (cit. on p. 10).

REFERENCES IN 3.2: SEMANTIC SCHOLAR (S2)

- [Sch19] Semantic Scholar. *Frequently Asked Questions*. 2019-05-23. URL: <https://www.semanticscholar.org/faq#paper-sources> (visited on 2019-05-23) (cit. on p. 11).

REFERENCES IN 4: FRONT-END

- [Fra+19] Max Franz et al. *cytoscape/cytoscape.js*. 2019. DOI: [10.5281/zenodo.831800](https://doi.org/10.5281/zenodo.831800) (cit. on p. 14).

REFERENCES IN 4.1: BACK-END

- [Lan14] Fabian Lange. *Why 35GB Heap is Less Than 32GB – Java JVM Memory Oddities*. 2014-02-26. URL: <https://blog.codecentric.de/en/2014/02/35gb-heap-less-32gb-java-jvm-memory-oddities/> (visited on 2019-05-22) (cit. on p. 19).
- [Luc12] Lucidworks. *Sizing Hardware in the Abstract: Why We Don't Have a Definitive Answer*. 2012-07-23. URL: <https://lucidworks.com/2012/07/23/sizing-hardware-in-the-abstract-why-we-dont-have-a-definitive-answer/> (visited on 2019-05-22) (cit. on p. 18).

REFERENCES IN 5.1: EXPLORATION

- [Kah+16] Minsuk Kahng et al. *Interactive Browsing and Navigation in Relational Databases*. 2016. arXiv: [1603.02371 \[cs.DB\]](#) (cit. on p. 22).
- [NSL18] Carolina Nobre, Marc Streit, and Alexander Lex. “Juniper: A Tree+Table Approach to Multivariate Graph Visualization”. In: *IEEE Transactions on Visualization and Computer Graphics* (2018). DOI: [10.1109/TVCG.2018.2865149](#) (cit. on p. 23).
- [Pie+17] Robert Pienta et al. “FACETS: Adaptive Local Exploration of Large Graphs”. In: *SDM*. SIAM, 2017, pp. 597–605. DOI: [10.1137/1.9781611974973.67](#) (cit. on p. 22).
- [SGC19] Mario Salinas, Daniela Giorgi, and Paolo Cignoni. *ReviewerNet: Visualizing Citation and Authorship Relations for Finding Reviewers*. 2019. arXiv: [1903.08004 \[cs.DL\]](#) (cit. on p. 22).

REFERENCES IN 6: METHOD

- [BH18] Maciej Besta and Torsten Hoefler. *Survey and Taxonomy of Lossless Graph Compression and Space-Efficient Graph Representations*. 2018. arXiv: [1806.01799 \[cs.DS\]](#) (cit. on p. 25).

REFERENCES IN 7.2: USER INTERFACE IMPROVEMENTS

- [Dwy+13] Tim Dwyer et al. “Edge Compression Techniques for Visualization of Dense Directed Graphs”. In: *IEEE Transactions on Visualization and Computer Graphics* 19 (2013), pp. 2596–2605. DOI: [10.1109/TVCG.2013.151](#) (cit. on p. 30).
- [HWR18] Mustafa Hajij, Bei Wang, and Paul Rosen. *MOG: Mapper on Graphs for Relationship Preserving Clustering*. 2018. arXiv: [1804.11242 \[cs.SI\]](#) (cit. on p. 30).

Name: David '-r' Schmid
Matriculation Number: 667815

Declaration

I hereby declare that this thesis titled:

**Combining Interactive Exploration
and Search for Navigating
Academic Citation Data**

is the product of my own independent work and that I have used no sources or materials other than those specified. The passages taken from other works, either verbatim or paraphrased in the spirit of the original quote, are identified in each individual case by indicating the source. I further declare that all my academic work was written in line with the principles of proper academic research according to the official "Satzung der Universität Ulm zur Sicherung guter wissenschaftlicher Praxis" (University Statute for the Safeguarding of Proper Academic Practice).

Ulm,

David '-r' Schmid