

ACQUISITION AND INTEGRATION OF PUBLIC DATA TO IMPROVE DETECTION OF SCIENTIFIC FRAUD

by

Ewoud Westerbaan

in partial fulfillment of the requirements for the degree of

Master of Science
in Software Engineering

at the Open University, Faculty of Science
Master Software Engineering
to be defended publicly on Friday April 1st, 2022 at 15:00 PM.

Student number: 852069942
Course code: IM9906
Thesis committee: dr. ir. Hugo Jonker (chairman & examiner), Open University
dr. Arjen Hommersom (examiner), Open University

CONTENTS

1	Introduction	2
2	Domain analysis of the publication process	5
2.1	Journal publication	6
2.2	Conference publication	7
2.3	Fraud in the publication process	8
2.4	Conclusion	9
3	Related work	10
4	Methodology	13
4.1	Method	14
4.2	Validation	15
5	Data model and standard data sets	16
5.1	Ideal data model	16
5.1.1	Conferences	17
5.1.2	Resulting model	17
5.1.3	Example: Publications cites publications in same venue	18
5.2	Publicly available datasets	19
5.2.1	DBLP	19
5.2.2	Aminer	20
5.2.3	OpenCitation	20
5.2.4	Google Scholar	20
5.2.5	Conclusion data availability	21
5.3	Integrability	21
5.3.1	Articles	21
5.3.2	People	22
5.3.3	Conclusion integrability	23
5.4	Processing of DBLP dataset	23
5.5	Conclusion	24
6	Generic Implementation	25
6.1	Data storage	25
6.1.1	Database	26
6.1.2	Transformations	27
6.2	Web scraping	27
6.2.1	Acquiring web data	27
6.2.2	Parsing HTML	28
6.2.3	Identity hiding	29
6.3	PDF scraping	29

7	Case study 1: Program Committee Members	31
7.1	Motivation	31
7.1.1	Interesting case: Work PC member is being cited	31
7.2	Implementation	33
7.2.1	Functional overview	33
7.2.2	Top conferences	34
7.2.3	Proceedings	35
7.2.4	Proceeding information, articles and authors	36
7.2.5	PC Members	37
7.2.6	Citations	43
7.2.7	Formalising the integration dataset	44
7.3	Analysis and validation	45
7.4	Case study conclusion	48
8	Case study 2: Journal editors	49
8.1	Motivation	49
8.1.1	Interesting case: Member of the editorial board is (co-)author	50
8.1.2	Case study purpose	50
8.2	Implementation	50
8.2.1	Formalising the dataset	53
8.3	Analysis and validation	55
8.4	Case study conclusion	56
9	Case study 3: Authors	57
9.1	Motivation	57
9.1.1	Interesting case: Author reviews his own work	57
9.2	Implementation	58
9.2.1	Journals	59
9.2.2	Articles	59
9.3	Analysis and validation	61
9.4	Case study conclusion	63
10	Conclusions	65
10.1	Future work	66
10.1.1	Data acquisition	66
10.1.2	Dataset integration	67
10.1.3	Alternative detection method	68
10.2	Recommendations	68
10.3	Discussion	69
	Bibliography	i
A	Dataset PC Member citations	iii
B	Research Proposal: Exploration of abstract graph-based approach to outlier identification	iv

ABSTRACT

Situation. The output of scientists is assessed by metrics like number of publications and citations. To turn these metrics to their benefit, various forms of fraud are being conducted. For fraud in publications, like plagiarism, various detection methods are in place. However, for post-publication fraud (fraud conducted after the actual production of an article) these detection methods are not structural in place and detection requires a lot of human effort. Improvement in detecting these types of fraud is to direct this human effort to the most egregious cases. However, generic approaches to detect these cases have failed because:

- Detection is applied on the whole population: a person can be outlier within a sub-group (e.g. editor targets its own journal, but other editors do not), but not within the whole population (other authors also target that journal);
- Public datasets these research are based upon, are too limited.

Research. In this research we investigate the added value of enriching existing publicly datasets with not yet integrated publicly available data for directing manual effort for fraud detection in the publication process. As validation we apply a group based outlier detection.

Main contributions. Our main contributions are:

1. Improved set-theoretic publication datamodel which can be used to reason about the publication model. Sources to load this model are discussed.
2. Acquisition methods to gather the necessary data to improve existing publicly available datasets. These enriched datasets can be used for fraud detection.
3. Case studies where we define sets gathered from public and additional acquired data. On these sets we apply a limited analysis to identify outliers. This is the proof our approach is actually working.
4. Recommendations to the research community which improves the detection of fraud on the publication process.

Key points. The results of this research are:

- For this study, it is not feasible to create one integrated dataset out of multiple publicly available datasets;
- Enriching standard datasets (datasets prepared for publication process analysis) is made more difficult because of diffusion coupled with lack of structure of data;
- Applying group based approach on enriched data yields useable results.

1

INTRODUCTION

Incentives for scientists rely on quantitative measures of science, such as number of publications, number of citations and amount of grants received. However, as the aphorism known as Goodhart’s Law states: “When a measure becomes a target, it ceases to be a good measure” [Str97]. That is the case here as well: these metrics not only incentivise scientific excellence by doing research and publishing the results; they also invite, maybe even more rewarding, dishonest approaches to achieve high rankings – gaming of these metrics, or, more simple, fraud in scientific publishing.

An interesting example where Goodhart’s Law could play a role is the following case: In 2010, Italy started using the number of citations as input for academic promotion. Seeber et al. [SCMM19] investigated whether this increased the number of self-citation. In Figure 1.1, Seeber et al. show the number of self-citations per year (the 2009 spike is not discussed). While the intuitive argument is clear, whether the data backs this up is subject to interpretation. It is clear from the graphs that the variance between various groups of academics is greater than the impact of the new rule. Data on the whole of Italian academia would therefore probably only show a slight increase. In reality, in certain subfields, the average number of self-citations would have more than doubled.

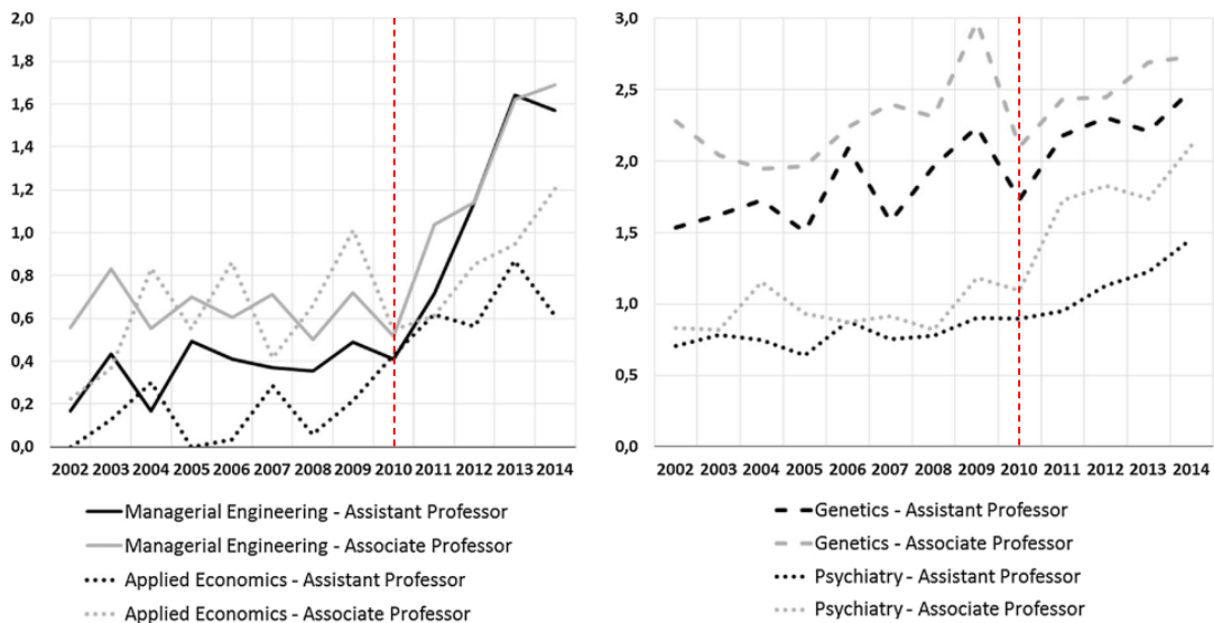


Figure 1.1: Evolution of self-citations per article (adapted from [SCMM19]). The red line indicates the moment citation-rates became input for promotion.

Other examples of fraud include plagiarism, manipulation of images (e.g., of cell extracts in life science publications), manipulation of research data, fabrication of research data. In addition, there may be as-of-yet unknown forms of fraud.

Various prevention and detection measures are in place. In absence of these deterrents, the scientific process would be vulnerable to dishonest behaviour. These existing deterrents are typically focused on specific forms of dishonest behaviour. Examples of these existing measures are Ithenticate to detect plagiarism, or ORI's Forensic Droplets¹ to detect manipulated images. The practice of preregistration ensures scientific studies are conducted as planned, and helps to prevent data manipulation such as p-hacking². Another measure is applying statistical evaluation to uncover manipulation of research data [HVWvA19].

Despite these deterrents, some forms of dishonest behaviour still slip through. Examples of such behaviour are: self-nominating as reviewer (e.g. Moon), forced citation (authors are being forced to cite work), forced co-author-ship (forced to name a person as co-author). Such behaviours use dishonest tricks outside the detective capabilities of current detection methods and therefore escape initial scrutiny. We know of such cases typically through whistle blowers and manual investigation. This illustrates that a problem still exists, yet the scale of the problem remains unknown.

In particular, one obstacle to prevent this problem is the huge (and growing) number of scientific publications and scientific authors. For example, the DBLP website, which records most publications in computer science, currently lists more than 2.5 million authors – a number that cannot be manually investigated. Another obstacle is the cost involved to investigate the possible scientific misconduct [MHWT10]. Hence, a more generic approach to detect fraud is needed.

¹<https://ori.hhs.gov/droplets>

²P-hacking is manipulation of the data so the research becomes statistically important.

In this project, we aim to take a step in this generic approach by not focusing on the method of fraud applied, but on the *effect of scientific fraud*, by taking a holistic view of the publication process. We tend to do this in two steps:

1. Provide a concise underlying integrated datamodel;
2. Identify outliers within groups sharing characteristics.

The underlying assumption this approach is based upon, is that the various types of scientific fraud all share a common characteristic: their goal is the same, that is, their goal is to improve a researcher's or publication's quantitative measures of science. This elevates them above their peers – which on the one hand brings recognition and accolades, but on the other, makes them stand out. In short, the underlying assumption is that fraudsters aim to become outliers (in terms of quantitative measures of science).

To clarify this with an example: An editor of a journal publishes in his own journal to boost his publication metrics. This may be fraudulent behaviour. If other editors of this same journal do not publish (or publish much less), this editor of interest is considered an outlier, because of the discrepancy with his group of peers. We immediately caution that the implication representing our assumption ($fraud \implies outlier$) does not hold in reverse. That is: such outliers need not be fraudsters: ($outlier \not\implies fraud$). However, identifying outliers may reduce the pool of authors to be investigated by several orders of magnitude. Thus, while not perfect, outlier classification should make manual investigation of cases where fraud would have had significant effect possible.

The goal of this project is to take a first step to make this a possibility. Concretely, our research objective is to integrate multiple data sources to enable the possibility of performing a group based outlier detection. The focus of this study is on the computer science discipline. For other disciplines the approach of this research probably needs to be tuned specifically. For example: the average number of co-authors per article change between research areas³.

During this research we will contribute the following items:

- *Improved set-theoretic publication datamodel* which can be used to reason about the publication model. Sources to load this model are discussed.
- *Acquisition methods* to gather the necessary data to improve existing publicly available datasets. These enriched datasets can be used for fraud detection.
- *Case studies* where we define sets gathered from public and additional data. On these sets we apply a limited analysis to identify outliers. This is the proof our approach is really working.
- *Recommendations* to the community which increases the detection of fraud on the publication process.

This thesis will continue with a domain analysis of the publication process in Chapter 2. After this background chapter, other related work is discussed in Chapter 3. The used methodology is described in Chapter 4. From here on we will answer the research questions which starts with the datamodel and - acquisition in Chapter 5, a generic implementation in Chapter 6 and follows with three case studies in Chapters 7, 8 and 9. We end this thesis with the conclusions in Chapter 10 with recommendations in Section 10.2 and discussion in Section 10.3.

³<https://www.natureindex.com/news-blog/paper-authorship-goes-hyper>

2

DOMAIN ANALYSIS OF THE PUBLICATION PROCESS

As stated in the introduction, incentives for scientists rely on quantitative measures of science, such as number of publications, number of citations and amount of grants received. The result of these measures being in benefit for the scientist are a higher status, salary and more resources (funds, students) to do more research which leads to even more rewards, recognition and accolades. To achieve these rewards, scientists need to have their papers published for their career to thrive. This process of publishing and rewarding is shown by Björk and Hedlund [BH04] in a cycle within their visualisation of the publication process. In Figure 2.1 we present this cycle.

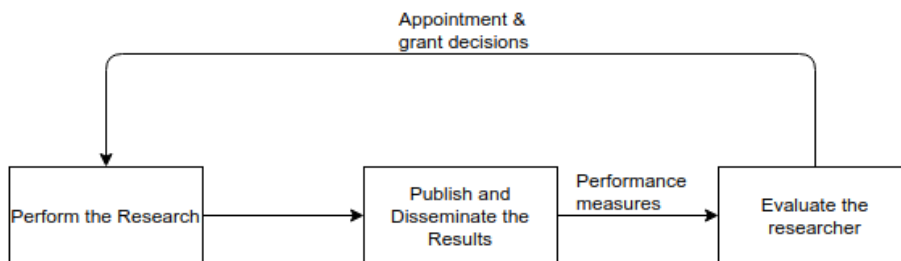


Figure 2.1: Publish for incentives (adapted from [BH04])

In this image we see that the researcher is evaluated and receives incentives (such as funds) based on performance measures. In their paper, Björk and Hedlund mention that for public bodies that provide funds (which is an incentive for the researcher), the production and consumption of publications should be optimized.

Two main targets exist for the production part, (the scientific publication): journals (incollection) or conferences (inproceeding). While journal publications are the norm in many other disciplines, conference publications are common in Computer Science.

The next sections describe these processes from a Computer Science perspective. For other disciplines these processes may differ. For example, in CS conferences, rigorous peer-review is the norm, with acceptance rates of 20% or less being common¹; in other disciplines, such strict peer review for conferences is less common.

¹Acceptance rates of security conferences: <http://jianying.space/conference-ranking.html>

2.1. JOURNAL PUBLICATION

First we discuss the process for publishing in a journal. Although the exact process may differ across journals, there is a certain common process. We tend to describe this common process.

The process of publishing in a journal starts with the author who has a manuscript he wants to publish. This can be based on a 'call for papers', or on own initiative. According to Cormode, the Editor-in-Chief receives the manuscript and does a minor check if the paper is good enough for further processing. If it is, the further process of handling the paper is assigned to an associate editor who also performs some checks for quality (understandability, duplication of prior work, topic in scope). The checks from the editor-in-chief and associate editor can lead to a rejection of the manuscript. According to Cormode, the author is encouraged to suggest an associate editor at submitting [Cor13]. This helps the editor-in-chief to delegate the handling to a associate-editor with knowledge of the domain e.g..

The main responsibilities of the associate editor are 1) initial handling and selecting reviews for the papers and 2) obtaining a decision for a paper. Although selecting reviewers to review a manuscript is a responsibility of the associate editor, it is not uncommon for an author to suggest reviewers. Sometimes the author even needs to provide reviewers.

After getting the results from the reviewers, the associate editor decides if the manuscript needs to be adjusted. Formally the reviewers give an advice, but it is up to the associate editor to do the final judgement. If adjustments are needed, the author adjusts his manuscript and this needs to be reviewed again. This can happen with other reviewers. An alternative path is that an author chooses to withdraw his paper.

Eventually, hopefully, if the paper is accepted by the associate editor, an advice will be given to the editor-in-chief. Formally the editor-in-chief decides if a paper is being published, but most of the time the advice of the associate editor is adopted. This chain of advice and assignment is shown in Figure 2.2.

Interesting in this schematic overview is the absolute power of the Editor-in-Chief. The person in this role can make a judgement without the advice of the associate-editor (and reviewers). Also this role can influence the process by choosing a specific associate-editor. Under the Editor-in-Chief, we draw the Associate editor. This role has two ways to influence the process; by choosing the reviewers and by giving the advice to the editor-in-chief. The reviewers have less power; of course they can provide an advice, but in most of the cases there are multiple reviewers.

To visualise the publication process for journals, we can use a model of Björk and Hedlund. They modelled the various processes involved in scientific publishing to investigate the business impact of the shift towards internet publishing models. This model is based on publication for journals. For our research, we took a diagram from their publication and took the activities usable for our research. Hereby neglecting the steps "Negotiate copyright" and "Copyedit Article". The result is shown in Figure 2.3.

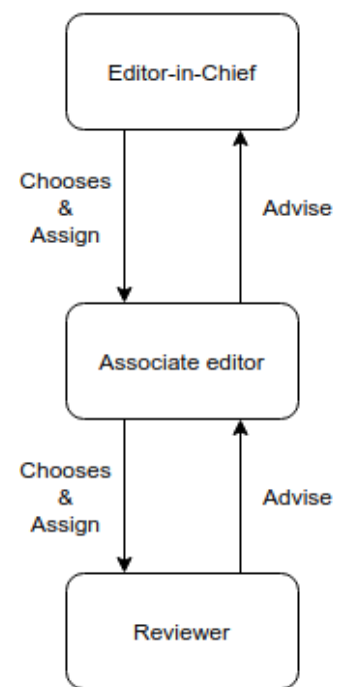


Figure 2.2: Roles within the journal process (visual interpretation of description in [Cor13])

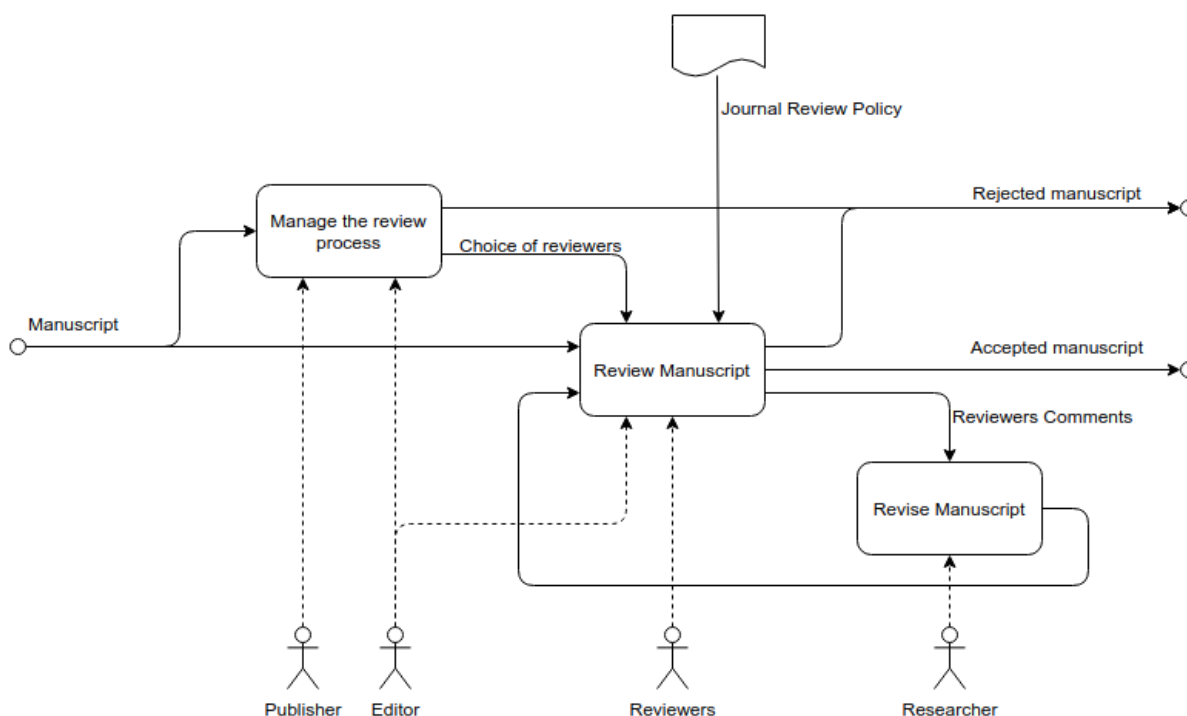


Figure 2.3: Journal process flow between submission and decision (adapted from [BH04, diagram A22331])

The phase "Manage the review process" from Björk and Hedlund is described by Cor-mode from the perspective of the Editor-in-Chief and Associate editor (Bjork and Hedlund decided to use one role: editor).

2.2. CONFERENCE PUBLICATION

The following is an informal description of the process for conference publication which can also be a workshop or symposium. Not all conferences follow this approach. Input for this description is provided by the supervisor of this research project.

For established venues the steering committee decides to have a conference and choose a location. The main responsibility of the steering committee is to oversee the organisation of the conference and to choose the Program Committee (PC) chairs whose responsibility is to compose a program out of a subset of the received submissions. For new subjects, the trigger comes from one or more academics who think a conference is needed for a certain subject and try to form a Program Committee (PC).

Possibly there are "program tracks" on specific subjects, with dedicated chairs for each track. The PC chairs invite people to become program committee members. A new trend is self-nomination; one can nominate himself for a position in the Program Committee. Another new trend is that authors of submitted papers are required to review. Besides inviting members, the PC chairs ensure the website is up, promotion is in place and have submission/review server set up.

The PC chairs create and distribute a call-for-papers, which starts the *submission phase*. Responding to the call-for-papers, scientists submit papers, and possibly marking conflict of interest. After a likely extension of the deadline, PC Chairs close submission and perform desk reject. After the submission phase closes the *bidding phase* begins. During this bidding phase, PC members bid on which papers they would like to review. In the *review phase*

the PC members read and score the papers. In this phase a paper can be early rejected with limited number of reviewers. Sometimes this phase contains a rebuttal phase; the reviews are sent to authors so the authors can reply on the given comments.

In the following *discussion phase* the PC members unify their views about the papers. After the discussion a decision should be made. During the *decision phase* the PC Chairs formalise a decision about a paper (formally; often they simply follow the consensus view). Possible decisions are accept, reject or conditionally accept (shepherding). In this case, the paper is interesting but not yet quite acceptable, salvageable. One reviewer is designated Shepherd and communicates with the authors on what changes are needed. Final acceptance hinges on the Shepherd officially approving the paper.

In the *camera-ready phase* the authors make final changes, incorporating editorial changes, reviewer comments, etc., and submit the final version for publication. In the *publication phase* the PC chairs judge on the the camera-ready version, and papers is published online.

2.3. FRAUD IN THE PUBLICATION PROCESS

In the introduction we already referred to the aphorism known as Goodhart's Law : "When a measure becomes a target, it ceases to be a good measure" [Str97]. Known metrics such as the H-Index (a performance metric for researchers) not only incentivise scientific excellence; they also invite other ways to achieve high rankings (e.g. plagiarism, manipulation of images, manipulation of research data, fabrication of research data).

This section provides short description of some interesting cases that may occur in the above described processes. It is certainly not the case that all situations do actually imply fraud, but these are cases that are a candidate for further manual investigation. Here we do not mention content-based fraud like plagiarism, we focus on fraud during the publication process. As Mario Biagioli calls them 'post-production misconduct', which is probably a nicer term [BL20].

Publications cites publications in the same venue. It is remarkable if a venue cites only work from itself. The benefit for this venue is that it increases the impact score of the venue. This case is described as an example in Section 5.1.3.

Work member of editorial board is being cited. Member of the editorial board that is excessively often being cited in his own venue, can be an indicator for fraudulent behaviour. This case serves as input for case study 1 (Chapter 7) and is described further in Section 7.1.1.

Member of the editorial board is (co-)author. A questionable situation occurs if a member of the editorial board becomes co-author of *relatively* a lot of publications. This may indicate that authors are being forced to add the member to the author list. The benefit for the member is an increased publication count and, if the article gets cited, citation count (which has impact on the H-Index and therefore impacts his career). This situation is described in case study 2 (Section 8.1.1).

Author reviews his own work. This kind of fraud is known because of Moon. Moon gave an e-mail address of himself (alias) as reviewer for his own work. This case was detected because of the short time between submission and accordance. Moon was detected, but are

other authors that show this fraudulent behaviour also detectable? Case study 3 (Chapter 9 is based on this publication-lag (time between submission and publication of an article).

Of course possible fraudulent behaviour is not limited to these cases. Other cases are described on the website of COPE (Committee on Publication Ethics)².

2.4. CONCLUSION

In this chapter we described the processes to publish articles for journals and conferences. In these processes fraud can be conducted, which we have shown with a few cases. The information in this chapter is domain knowledge which is needed to conduct this research.

²<https://publicationethics.org/guidance/Case>

3

RELATED WORK

As stated in the introduction a more generic approach for fraud detection is needed. This research is a continuation of the research of Tielenburg, which tried to apply this generic approach [Tie17].

Tielenburg investigated automated detection of fraud-like behaviour based on publicly available data gathered from DBLP and Google Scholar. He determined fraud indicators based on a data model of the publication process proposed by Jonker and Mauw [JM17] shown in Figure 3.1. In this model V , P and A stand for Venue, Paper and Author.

Tielenburg proposed three heuristics calculated about the authors to identity outliers: *maxdiffpubcount* (maximum over the difference in the number of publications between two consecutive years), *maxdiffcitecount* (maximum over the difference in the number of citations between two consecutive years) and *maxratioctitsvspubs* (maximum of the ratio between citations and publications). The latter is a derived value based on publication and citation counts per year.

Tielenburg's implementation based on the data modelled by Jonker and Mauw was not successful; entities (e.g. the reviewer) are missing in this model. Therefore, this model needs to be extended.

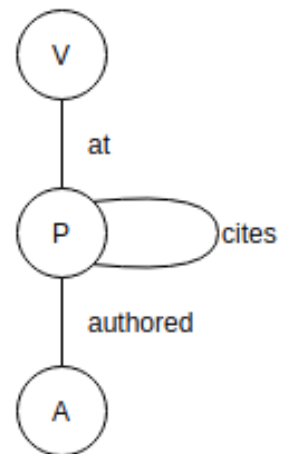


Figure 3.1: Set-theoretic publication model adapted from [JM17] used by [Tie17]

Extending the model. Haug raised attention to fraud in the peer-review process and the important role (guest-)editors have in relation to the peer review process [Hau15]. The attacks she presents are self reviewing like Moon and Chen and fake reviewers. Another attack she mentions is the attack on the systems supporting the peer-review process.

Bishop¹ additionally considers the impact of the role of the editor. In her blogpost she raises attention to members of the editorial board which are also co-authoring articles. However, this is mainly focuses on one editor.

Scanff et al. [SNC⁺21] focused on the publication process of prolific authors, who might be part of the editorial board. They measured the publication time (time between submis-

¹<http://deevybee.blogspot.com/2020/07/percent-by-most-prolific-author-score.html>

sion and acceptance) of these prolific authors and noticed how this period is less for papers written with prolific author compared to papers written without prolific authors.

Available datasets. *DBLP* is a publicly available dataset focused on publications in the Computer Science Domain. The dataset contains publications, authors and venues. Ley describes the choices made by DBLP composing the dataset [Ley09]. The person entity is described in detail included the procedure DBLP takes to identify unique people with the same name or an author with multiple names. The data DBLP uses to create this dataset is being pushed from publishers or scraped from the web using web crawlers. DBLP covers the datamodel of Jonker and Mauw without the citations.

Heibi et al. introduced the *OpenCitations* COCI (Crossref OpenCitation Index) dataset. This dataset contains citations on DOI-to-DOI based [HPS19]. OpenCitations acquired their data from undisclosed datasources.

Aminer is a service that provides datasets for analysis of the publication process. Tang et al. described how *Aminer* loads the source data from sources and uses probabilistic models to deal with name disambiguity [TZY⁺08]. As with DBLP, Aminer also elaborates on how they tackle the author-naming problem. The citation dataset, which contains authors, publications and citations, uses data from DBLP, ACM and ‘other sources’. Aminer covers the model from Jonker and Mauw, focused on citations. For the entities involved, a lot of attributes are not available.

These datasets are not complete. Chakraborty et al. [CPN21] found data on the various roles in the publication process lacking to such an extent, they could not complete their originally envisioned research. Thus, while specific roles may significantly impact the publication process, data regarding these roles is not readily available. Data acquisition is thus a key component of any study into the correlation between specific roles and impact. Therefore, acquisition of additional data is needed.

Data acquisition and parsing. As described in previous paragraph, the actual source data used to create public datasets is not transparent. However, generally we notice that the most ‘original’ data (not acquired from other public datasources) is acquired from websites of publishers (e.g. using the webcrawlers in case of DBLP). Besides websites, publications and editorials are most of the time available in PDF format. Therefore we focus in this paragraph on acquisition which contains ‘downloading’ the content (which may be HTML or PDF) and parsing, transforming the acquired data in information.

For downloading content from the web (e.g. HTML or PDF documents) standard tooling exists (e.g. `cURL`², `wget`³). For incorporating this downloading in software, multiple libraries exist. In case of HTML two types of methods for acquiring content exist: web browser wrappers (e.g. Selenium) and HTTP libraries (e.g. `Urllib2`) [Zha17]. The first simulates a browser. The advantage is that dynamic content can also be acquired. The second is a single GET request (which is equivalent to tooling described before). The content acquired differ from site to site. Therefore, this content needs to be interpreted to turn this acquired data into information.

Parsing is extracting information from the acquired (raw) data. The parsing mechanism de-

²<https://curl.se/>

³<https://www.gnu.org/software/wget/>

depends on the type of data. Examples are HTML (web pages), datafeeds in XML and JSON or multimedia types. Considering the type of data his research is conducted on, we focus on HTML and PDF files. Zhoa [Zha17] mentioned BeautifulSoup for interpreting HTML and Pyquery for XML. Uzun et al. [UYK18] compared regex and DOM-based libraries (Lxml and BeautifulSoup) for data parsing. DOM-based libraries were better for parsing than using regex. Lo et al. interpreted PDF's from multiple papers [LWN⁺20] to extract bibliographic information. They also tried to parse Latex documents, but surprisingly, the results were worse with Latex than with PDF. For extracting information from PDF's, they relied on a project called ScienceParse⁴. This is framework that interprets Scientific Papers. The main component used to read the PDF documents, is PDFBox. This parsing of PDF's is similar to [NX14] and [BK17]. Bast and Korzen performed a benchmark of tools for extracting text from PDF documents. Considering the error-rate of various tooling, pdftotext, pdftohtml, pdftoxml, pdfbox and parscit are considered the top 5 tools [BK17]

Modelling the publication process. For integration of multiple datasources, we need a model of the publication process. We already mentioned the set-theoretic publication model presented by Jonker and Mauw [JM17], shown in Figure 3.1. This model is used to formulate an attack surface on the publication process. However, based on work from Tielenburg, Haug, Scanff et. al. and Chakraborty et al. we can conclude that this model does not represent the necessary information.

Björk and Hedlund [BH04] describe a model of the publication process for measuring costs. Their model includes a large set of involved entities and roles. While this model provides a solid foundation for understanding the publication process, the model is a process model and does not facilitate deriving publication metrics. As such, it is unsuited for determining the value of quantified fraud indicators.

From the view of the associate-editor, Cormode describes in detail how his work looks like [Cor13]. It provides a good insight of the work of an associate editor, which can also be read from an 'attacker-mindset' (where are the vulnerabilities), but unsuited for fraud detection.

Data integration. After acquisition and parsing, the data needs to be integrated. This integration is a common step in ETL (Extract Transform Load). ETL is known as a process used in data warehousing (component to serve historical data formatted for reporting and analytics). Therefore, most integration techniques come from this area of expertise. Physical datamodelling techniques are Data Vault [LO15] and Anchor modelling [RRB⁺10]. Both are ensemble modelling techniques; this means focusing on a business level (business entity based) and an agile approach (the model can be changed and grow incrementally). An important aspect of these techniques is the availability of a cross-source key (business key) to match the same objects from different sources. These physical datamodelling techniques can be used in the implementation phase of our study, if such cross-source keys can be identified and the quality is acceptable. For example, in the domain of scientific publication, this role could be filled by DOI⁵ or ORCID⁶, if used in all sources.

⁴<https://github.com/allenai/science-parse>

⁵Digital Object Identifier, an identifier intended to uniquely identify a scientific publication.

⁶Open Researcher and Contributor Identifier, an identifier intended to uniquely identify people involved in scientific publications.

4

METHODOLOGY

In Chapter 2 we described the processes to publish articles for journals and conferences. In these processes fraud can be conducted. To improve detection of this fraud, data about these processes is necessary. Previous approach ([Tie17]) to discover fraudulent behaviour was based on public datasets, but misses important entities. Considering the cases described in Section 2.3 and public datasets that are generally represented by Jonker and Mauw [JM17], missing entities can be pointed out. In Figure 4.1 the generic data model for public sources is on the right side; V for venue, P for paper and A for author. Left of this model some missing entities are added.

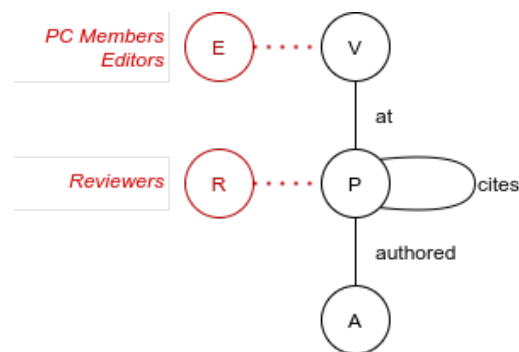


Figure 4.1: Right: Set-theoretic publication model adapted from [JM17] Left: Missing entities considering example attacks.

However, these missing entities are available, but not in a structured format and integrated in existing datasets. Our assumption is that integrating additional data will improve directing manual effort in detecting fraudulent behaviour. This brings us to the main question of this research:

To what extend can integration of publicly available data sources contribute to directing manual investigation of scientific fraud?

4.1. METHOD

On a high level, this is a two step process:

1. Define the datamodel for the publication process;
2. Acquire and integrate data to fill this datamodel.

Define datamodel. Missing entities are mentioned in the problem described in the lead of this chapter. The resulting figure (Figure 4.1) is not sufficient to build a dataset for fraud detection. Therefore, as first step a more concise datamodel is needed to enable detection of interesting cases. With this step we identify entities and their relations. Possible approaches to define a datamodel are:

- *Use an existing datamodel:* Datamodels about the publication process do exist. An initial analysis made clear that existing models are somehow limited to the author, venue and publication. Attacks as described in Section 2.3 are not possible to detect using these models. As an example: the Microsoft Academic Graph model¹ (which is the most detailed we came across) does not contain any other role than author.
- *Create a model:* The domain analysis can be used as input to create a datamodel.

A combination of these approaches will add the most value for this study. The reason is that existing models are a good starting point. However, some extensions are needed for our study. We scope this step by modelling a logical model. There are two reasons for this:

1. A physical model containing all possible attributes is too detailed. Taking this approach will take too much time while the added value is not worth the effort.
2. By using a logical model, the focus is on the entities and their relations. These are the most important components for a group based detection.

This does not neglect the need for a physical model for implementation. Multiple physical modelling techniques exist which can be used for implementation.

Acquiring and integrate data. The second step is to determine which publicly available dataset can (partially) fulfill some parts of the model defined in the previous step. Our approach is here to discuss the available datasets and how these cover the model. Based on the research preparation, we know DBLP and Aminer are two sources that we probably can use. In this step a more extensive investigation is needed focused on integration and covering entities and relations from the resulting model from the first step. Sources for required entities not covered by public datasets should be found. For every source we should investigate the following points:

- How the acquisition will take place. This depends on the source.
- How integration of this newly acquired data should take place. This depends on available attributes.

An approach for these added sources can not be decided up front.

¹<https://docs.microsoft.com/en-us/academic-services/graph/reference-data-schema>

4.2. VALIDATION

As we know from our research proposal, publicly available datasets do not contain necessary information to identify groups. Therefore, data to apply group identification needs to be acquired from additional sources and integrated with other existing datasets. Detect outliers within such groups which can be used to direct manual investigation, proves the added value of integrated data.

Apply group based outlier detection. In this step we define groups and apply a group based detection on the dataset composited from the previous step. In the domain analysis of the publication process interesting situations are described with the groups having the power to abuse that particular situation (Section 2.3). In Figure 4.2 this is these are represented with the arrows from the publication process to the roles.

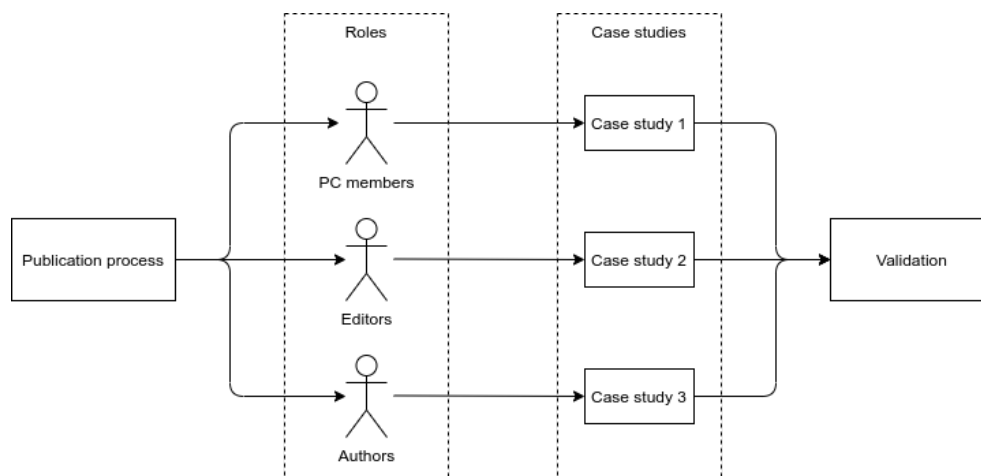


Figure 4.2: Relation publication process, roles and case studies

Based on these situations three cases studies (identifiable groups) are formulated:

- Program Committee Members
- Editors of a journal
- Authors based on publication lag

For every use case we do the following:

- Create an analysis dataset from the previous step focussed on this group and its metrics;
- Validate possible interesting cases.

Formalising the conclusion. The combinations of validations from the case studies will result in the answer on the question to what extend the integration of publicly available data sources contributes in directing manual investigation of scientific fraud.

5

DATA MODEL AND STANDARD DATA SETS

In this chapter we describe the underlying data model to perform group based outlier detection. This chapter addresses three points:

- The ideal data model for the publication process;
- Known data sources to fill this data model;
- Completeness of these known sources to fill the data model.

5.1. IDEAL DATA MODEL

By formalising the ideal data model, we can answer which entities and which roles are involved and their relationships. The ideal situation is one dataset which contains all entities and relations and can serve as base to easily create the necessary datasets upon to perform group based data analysis. As input to reason about this dataset we use the domain analysis from Chapter 2. As starting point: Jonker and Mauw formalised the publication model [JM17] and created an entity based view of the publication process. Figure 5.1 depicts their final model.

Whereas the Jonker and Mauw only modelled the author as a role in the process, Björk and Hedlund mention the publisher, editor and reviewer as well. We consider the author from Jonker and Mauw the same as the researcher from Björk and Hedlund. In this research, we ignore the publisher. Although the publisher is responsible in the end and sets the regulations and constraints for editors to work within, the actual work is delegated to editors.

Extending the model of Jonker and Mauw with the description of Cormode [Cor13] results in a model that incorporates the most important roles with influence on the judgement of the work of an author. We ignore roles like the editorial assistance and subeditor, because these roles do not seem to have influence on the judgement of a manuscript.

In Figure 5.2 we provide an overview of the entities and roles from Jonker and Mauw combined with the information from Cormode which answers the roles involved and their dependencies. The red rectangles are objects and the green circles are roles. As objects we have a venue which contains papers. Multiple roles are related to these objects.

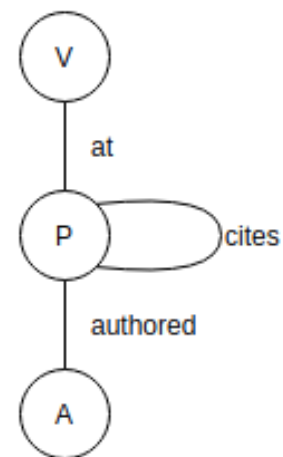


Figure 5.1: Set-theoretic publication model of papers (P), venues (V), and authors (A) (adapted from [JM17])

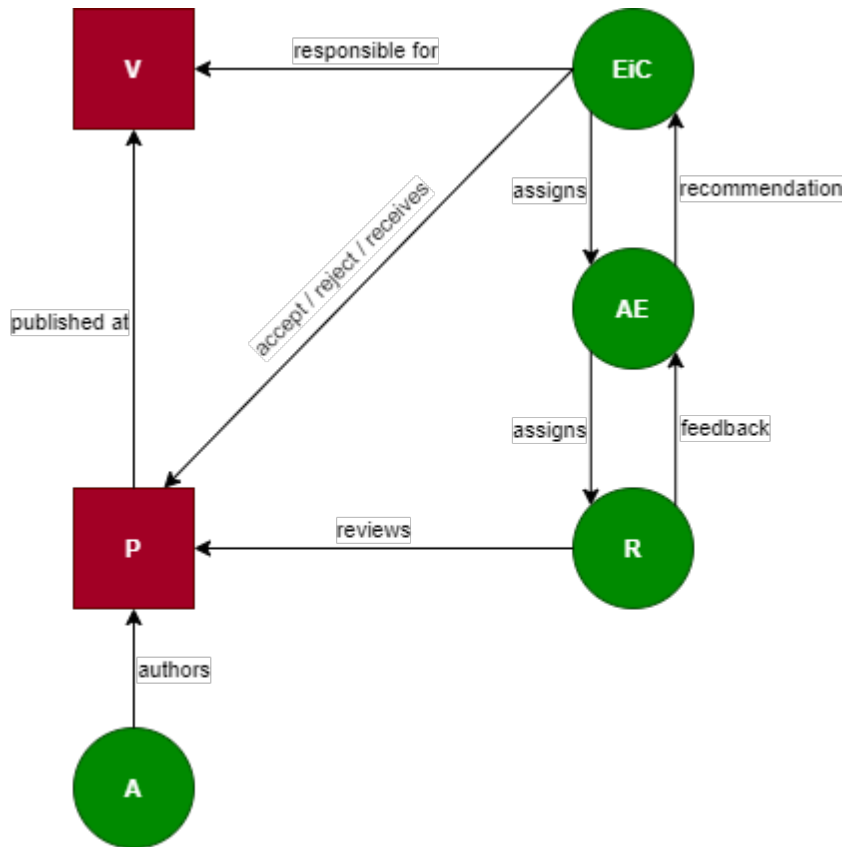


Figure 5.2: Set-theoretic model of the publication process for collections.

It is important to notice that the green circles are roles, not entities. These roles are fulfilled by people. This results in one more entity (person) with relation to other entities. These relationships can be expressed as the roles (e.g. a person is a reviewer for a publication).

5.1.1. CONFERENCES

So far we only focused on the process of journals. But, as mentioned before, publishing for a conference is important in the Computer Science domain. In case of conference publications, the author, publication and its relation to the venue stay the same. It is possible that this venue becomes more extensive because of tracks, but it stays a venue. An important person for conferences is the Program Committee Member. Eventually this is again a role of a person related to the venue.

5.1.2. RESULTING MODEL

In the resulting model we choose to abstract the roles away and use one entity called person where we define the roles to certain entities in the relationship. This makes the model far more easy to reason about when a person has multiple roles.

The model in Figure 5.3 is a logical model. A physical model can be designed based on this logical model using design strategies like Ensemble Modelling (e.g. Data Vault [LO15] or Anchor modelling [RRB⁺10]). Which modeling principle is being applied is not important. What is important, is that the physical model address requirements like time-

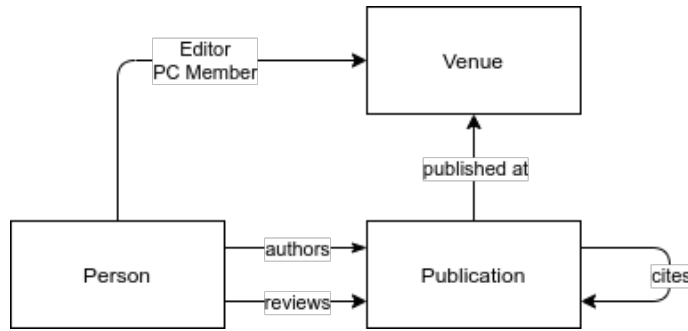


Figure 5.3: Set model which covers necessary entities and relationships

awareness and tracking changes of entities to create a physical model. E.g. a person is not 'always' editor of a venue, this role starts and stops at a certain moment in time.

5.1.3. EXAMPLE: PUBLICATIONS CITES PUBLICATIONS IN SAME VENUE

In this part we will go through a situation to describe the process of analysing a situation which can be an attack on the publication process and plot this on the resulting datamodel from the previous section. The chance this particular situation occurs is not very high. The reason is that there is a good enough safety net to catch this situations provided by Thomson Reuters¹, institute that calculates the Journal Impact Score. Besides, in our research we are looking for interesting people. Because this description method of an interesting case is used later in this thesis, description of the steps taken are added.

First we describe the situation and provide a model of instances that are involved. This can be a person in a certain role, venues, publication etc.

Having publications citing publications from a certain venue, increases the journal impact factor. Figure 5.4 is a model of this situation; a venue (v1) that contains a publication (p1) which has a citation (c1) to a publication (p2) in the same venue. As an example to clarify the notation, p1 and p2 are both different instances of a publication; we will use this notation when we describe other cases as well.

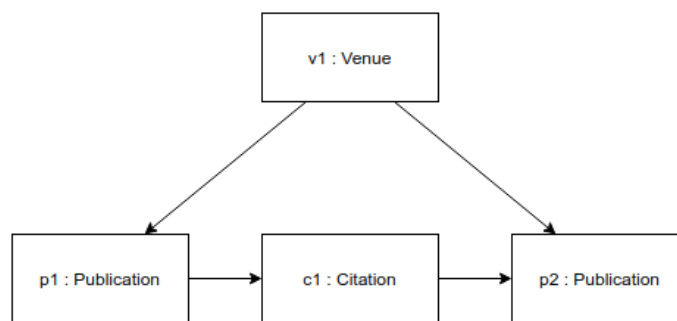


Figure 5.4: Instance model of citation of publication in the same venue

Attack. *We describe when (this is not always the case, see 'Benign alternatives') and how this situation is an attack.*

¹<https://clarivate.com/webofsciencegroup/solutions/journal-citation-reports>

It is not unusual that publications in a journal cite publications from the same venue. However, as with most attacks, when this occurs much more compared with other venues and the 'inner venue'-citation rate is significant higher, it becomes interesting. This may indicate that this is a conscious action to increase the Journal Impact Score, and this can be considered an attack.

Model impact. *In this part we describe how this situation impacts the publication model. What does in- or decrease and how does this impacts the 'score' of the attacker.* The impact on the publication model is that the number of citations to publications in the same venue raises. Therefore, the impact score of a journal raises.

Benign alternatives. *In this part we describe how this situation can occur, while not being an attack.*

If a certain venue is that specialised in a certain topic, the situation where a publications cites other publications in that venue can occur.

5.2. PUBLICLY AVAILABLE DATASETS

In this section we will explore some publicly available datasets with the focus on Computer Science which can be used to fill the model from Section 5.1.2. Because of the necessary ability to integrate datasets, we will judge these sets on integrability and completeness. As a guideline, we will use Figure 5.5. Our main target is to create a publication view based on the data sources which are represented in the figure as 'raw data'.

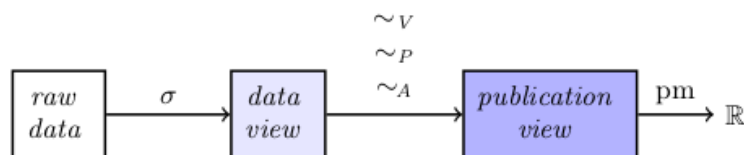


Figure 5.5: Data flow ([JM17])

The sigma-sign represents the filter on the set. For now, this filter sets the focus on the Computer Science area. Later on, this filter can be more strict, depending on the case study. The tilde-sign is to combine multiple representations of the same object from sources to one object (e.g. a paper available in two different datasets is represented by one paper in the publication view). Furthermore, the pm is a 'publication metric'-function resulting in metrics (e.g. number of citations).

A known datasource for Computer Science in DBLP. We can use this dataset as base. For other research areas, other datasources should be taken (e.g. for biomedical research, one can use PubMed).

5.2.1. DBLP

DBLP is a dataset with bibliographic information in the computer science discipline which can be downloaded in XML format. In Figure 5.6 we show which elements of the set are covered by DBLP.

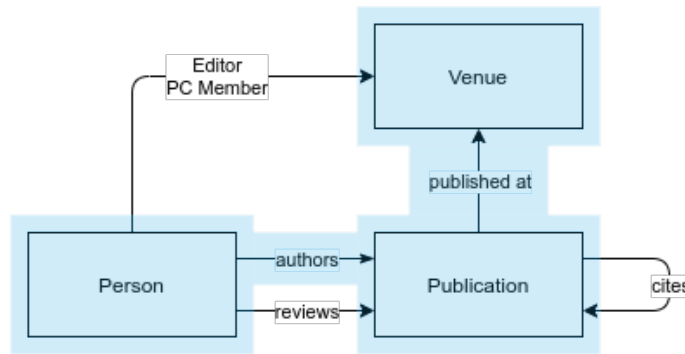


Figure 5.6: Set model DBLP

Unfortunately DBLP misses some important information to be complete; the dataset does not contain references, editors, PC members and reviewers. For references, other sources can be addressed: Aminer, OpenCitations or Google Scholar.

5.2.2. AMINER

Aminer positions itself as an AI for academic publications [TZY⁺08]. One dataset they provide is the 'Citation Network Dataset'. This dataset combines data mainly from DBLP, ACM (a publisher) and MAG (Microsoft Academic Graph). Because it uses DBLP as a source, it is interesting to investigate if we can use this for filling the gap of the citations shown in Figure 5.6. In Figure 5.7 we presented which entities and roles from our ideal data model this dataset contains. Although this set contains data from these entities, the attributes are limited; the main purpose of this set is citations.

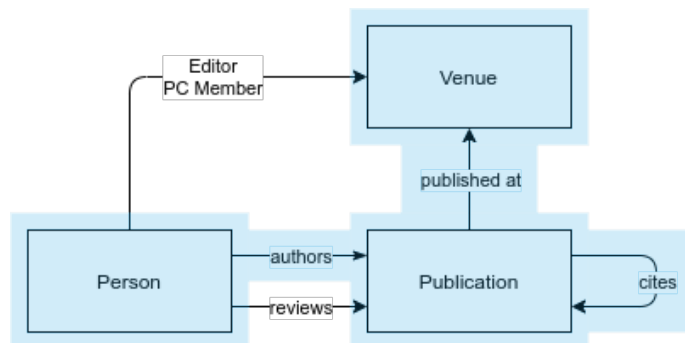


Figure 5.7: Set model Aminer

5.2.3. OPENCITATION

OpenCitation is a DOI based reference dataset. This means, the cited and citing articles are referred to by a DOI. The dataset from opencitation can be downloaded, or questioned by an API request.

From the publication entity it only contains the DOI.

5.2.4. GOOGLE SCHOLAR

Google Scholar also contains the references from and to articles. This data is only available online. Some software packages exists that support scraping this information. However, experiments from the research preparation phase concluded that this was not a feasible

solution because of the amount of captcha's when trying to scrape the site. The acquisition of this data is not practical.

5.2.5. CONCLUSION DATA AVAILABILITY

Referring to the ideal model as given in Figure 5.3, we can make a few statements:

- The availability of data about publications, venues and authors is good; almost all public sources contains some information about these entities.
- References are available in two publicly available sources (we ignore Google Scholar for given reasons in Subsection 5.2.4).
- Other roles than 'author' are not available in public datasets.

In the next section we will investigate how these sources can be integrated.

5.3. INTEGRABILITY

People, articles and venues can be acquired from multiple source. To be able to work with these entities, we need to integrate these sources in one model. To make this possible, we need to uniquely identify the entities. In the ideal situation entities have an unique identifier which is independent of the source. However, in practice it is a bit more complex than that.

5.3.1. ARTICLES

The unique key which is used to identity an article is the Digital Object Identifier. This field is available in DBLP, Aminer and OpenCitations. Also, the DOI should be unique across all articles. Theoretically, combining the sets using the DOI should be possible without much effort. However, an analysis of the DBLP data set leaves us with a few DOI's which refer to multiple articles. The number of articles this occurs, is very low (approximately 20 articles; a manual override is a possibility). Also, DBLP has a lot of documents where the DOI is unknown; so data completeness is an issue here.

DOI AVAILABILITY

DBLP contains additional information about an entry in an ee tag. These contain links to doi.org, wikidata, etc. Sometimes an article has multiple doi.org links; this seems mostly the case with the publisher ACM. Although the links are different, they redirect to the same page at ACM. For articles the following count of DOI's is available.

# of DOI	# of articles
0	1,051,229
1	4,626,193
2	22,396
≥ 3	1,202

Concluding; most articles do have a DOI; approximately 78% has one or multiple DOI's.

Aminer provides one DOI field for an article. Also not all DOI's are available. For the 4894081 articles in Aminer, 3920939 has a DOI (approximally 80%). However, the DOI's provided in the dataset are not unique: 3316 DOI's are used to identify multiple articles. A

quick investigation taught us that Aminer sometimes uses a DOI to refer to a journal instead of an article. Another finding with a quick analysis in the case of DOI's, is that multiple articles are represented multiple times in the dataset; the article dataset is not unique. This raises some doubt about the quality of the dataset of Aminer.

ALTERNATIVE KEY

Because the DOI is not a useful key across all sources to integrate data because of availability and uniqueness, alternative fields that make a good candidate to serve as key are looked into. A combination of title, authors, year and journal makes a good candidate.

The title and how sources save this title may differ. Sometimes the sources uses other characters. For example, DBLP uses an HTML tags to specify characters in the title, e.g. see Listing 5.1. This in combination with specific symbols (e.g. Greek delta-sign) which are expressed differently depending on the source, makes joining sets on titles with these signs cumbersome.

Listing 5.1: Example title in DBLP

```
<title>Graphs of Bounded Treewidth can be Canonized in AC1.</title>
```

These problems can be overcome by applying some clean rules before the join. Considering a solution for the title exists, still a join on other attributes needs to be done; the title itself is not unique. Which raises a next integration problem: people.

5.3.2. PEOPLE

In the publication domain an identifier for a person exists, this is called the Open Researcher and Contributor ID (ORCID). This field is (partially) available in DBLP, but not in Aminer. Alternative approach to uniquely identify authors is the name. However, this has multiple issues:

- *Format*: A name can be written in multiple formats. E.g. a person is named Alfred Jodocus Kwak; this can be written as 'Alfred Kwak', 'Alfred J. Kwak', 'A. J. Kwak', 'A. Kwak' and of course the full name. With names that contain accents on letter, the possibilities increases. Also, in some countries it is used to put the family name first.
- *Multiple people with same name*: The name does not make an author unique. The same name refers to other authors.
- *Other names for the same person*: E.g. when Robert is also called Bob.

Both DBLP and Aminer acknowledge this problem. In case of multiple people with the same name, DBLP addresses this problem by suffixing the name with a four digit code². DBLP is making progress here, but this is not entirely done yet. The format issue and alias issue is covered by DBLP by creating a master record for the author, see Listing 5.2 as an example. Aminer uses an approach which takes the coauthors in to account. It creates a network of co-authors for a certain author and then calculates a probability that this is the same author.

However, an exploratory analysis of the DBLP data makes clear this is not flawless; cases exist where the same master record authors one article multiple times. After a small investigation it seems that a father and son both authored this article (Robert and Bob, same lastname so DBLP 'chooses' to treat these names as the same person).

²<https://dblp.org/faq/1474704.html>

Listing 5.2: Example master record(<https://dblp.org/faq/1474690.html>)

```
<www key="homepages/r/CJvanRijsbergen">
<author>C. J. van Rijsbergen</author>
<author>Cornelis Joost van Rijsbergen</author>
<author>Keith van Rijsbergen</author>
<title>Home Page</title>
<url>http://www.dcs.gla.ac.uk/~keith/</url>
</www>
```

But hypothetically, even if DBLP manage to unique identity authors, the match to authors in other datasources is not possible, if these sources do not adhere the same method.

5.3.3. CONCLUSION INTEGRABILITY

From this section we can draw the following conclusions:

- Attributes that could serve as a cross-source key to integrate data (ORCID, DOI) are not complete enough;
- Alternative methods to integrate the data have too many weak spots to serve as a sustainable solution;
- Therefore, integration of datasets is cumbersome.

Combining these statements with the statements from Subsection 5.2.5, we choose DBLP as root for the dataset and choose to add additional data depending on the case study.

Therefore, for the case studies a pragmatic approach should be taken; determine what data we need to incorporate, what attributes are at our disposal and make choices based on that information.

Next step is to actually load the data of DBLP.

5.4. PROCESSING OF DBLP DATASET

The dataset that can be downloaded of DBLP is XML based. For every bibliographic record the file contains a XML element. These record contain elements with additional information, e.g. author, title and journal. In Listing 5.3 we show one bibliographic record (an article) as delivered in the dataset of DBLP.

Listing 5.3: DBLP bibliographic record example

```
<article mdate="2019-10-25" key="tr/gte/TM-0332-11-90-165" publname="informal">
<author>Frank Manola</author>
<author>Mark F. Hornick</author>
<author>Alejandro P. Buchmann</author>
<title>Object Data Model Facilities for Multimedia Data Types.</title>
<journal>GTE Laboratories Incorporated</journal>
<volume>TM-0332-11-90-165</volume>
<month>December</month>
<year>1990</year>
<url>db/journals/gtelab/index.html#TM-0332-11-90-165</url>
</article>
```

Possible approaches to load this dataset are:

- Option 1: Load all elements to separate tables** This approach splits all the data in separate tables; it converts this semi-structured dataset into a structured (relational) dataset. The positive points are that we have all data (probably also data we do not need, but we do not know that at forehand) and it is a process which can be automated. Also, storing the dataset in a relational database gives the opportunity to integrate the data with other sources. However, this approach leads to a lot of tables, cost storage and requires a structure that keeps the referential integrity.
- Option 2: Keep it as a file and make software to question the file** We can choose to not load the data at all, and create some interfacing to the file to be able to read the data. By taking this approach we need to know the model of the data at forehand and create software that gives access to the data in the file. Also, we are not able to easy query the data, unless we build this functionality; functionality which already exists in every other database. However, taking this approach does not need to copy the data into another structure, which is positive for disk allocation.
- Option 3: Split the file in separate XML's and load these in the database** Nowadays, databases are able to handle a XML structure as a native datatype. The drawback is that, even current relational databases support this task, the performance is not optimal. Also querying this data is more complex and slow compared when all data is relational stored. Using another type of database is also an option (document based). However, for analysis tasks we are going to perform, this is not an ideal solution. These document databases are built for 'single' requests (e.g. to support an API with predefined requests), and not for analysis workloads (e.g. aggregating).

The approach we take is to load all subelements in separate tables (option 1). The main reason is the possibility for integrating this set with other sources. Given the example in Listing 5.3, this results in multiple tables: Article, which is the main object, Author, where all authors can be put and separate tables for title, journal, volume, month, year and url.

5.5. CONCLUSION

In his chapter we defined the model which we need to fill to be able to analyse the publication data. Also, we explained how we can fill this model with publicly available data sources and what elements are missing. From the sets we discussed, none of them contains data to load the relation between person and venue (e.g. Editor and PC Member). In the next chapter we will describe some generic implementation strategies. After this, we will work out three case studies and dive deeper in the missing data and how we can acquire this.

6

GENERIC IMPLEMENTATION

Working with standard datasets solely is shown not to be successful for detecting outliers [Tie17]. Therefore other sources should be addressed. Some data about the publication process is available on websites of publishers or in PDF documents (e.g. Front Matter documents and editorials). Therefore, software is needed to acquire data from the web and from PDF documents. Besides acquisition, this data needs to be stored. This chapter is case-independent; we describe some generic components, but case specific software will be described in the chapters for these use cases.

We start with discussing data storage, then transformation of datasets. After those data related sections, we continue with discussing scraping data from the web and some generic information about PDF parsing.

6.1. DATA STORAGE

During this research project we acquire data with different formats (e.g. JSON, XML, HTML, CSV, PDF). In general, our approach is to split this process in *ingestion* (storing the data in raw format) and *interpretation* – though this approach cannot be applied in every case. Splitting the process invokes a separation of concerns; there is no need to re-acquire the data when interpretation fails. To store this data in its original format we need disk storage. After interpretation, data needs to be stored in a way that supports transformation (e.g. cleaning), integration and analysis. For this purpose, we choose a relational database.

This process is shown in Figure 6.1. As an example, the data source can be a web page. The acquisition process acquires the raw HTML and stores the data on disk. An interpretation process interprets this data (converting data to information) and loads the data in the database; ready for transformation, integration and analysis purposes.

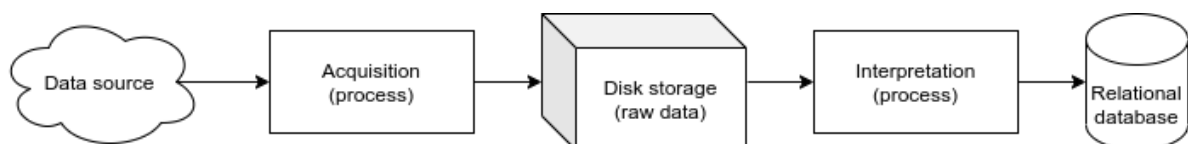


Figure 6.1: Storage separation

Risks. Storage comes with risks, e.g.:

- Risk of failing infrastructure;
- Risk of having not enough storage.

Both risks can be mitigated by using services supplied by a company specialised in running and maintaining infrastructure and services: the cloud. By using cloud service, backups are automatically created, in case of a database; we could go back to a certain snapshot and the responsibility of keeping the services running and up-to-date is delegated.

As cloud service we use Microsoft Azure. The reason we choose Azure is because we have free credits for this cloud provider; we did not find another reason to choose one cloud provider over another (e.g. Amazon, Google); all cloud providers offer disk storage and relational database services.

6.1.1. DATABASE

Microsoft Azure offers open-source types and (their own) proprietary relational database (SQL Server). There are two reasons not to choose the proprietary Microsoft SQL Server:

- The costs of proprietary exceeds open-source database alternatives and our cloud credits are limited;
- The proprietary SQL Server contains functionality we do not need (e.g. row-level security, depending on your rights you can see certain roles).

Choosing between the open-source types (MySQL, MariaDB, Postgres) is a personal preference: this project uses a Postgres¹. Other cloud providers offer the same service (e.g. Amazon RDS for Postgres² or Google Cloud SQL for Postgres³).

ABSTRACTING LOADING DATA IN A DATABASE

A relational database is schema-on-write. This means, you first need to define the structure of the data by creating a table, before data can be written to the database. Creating tables and adding new columns in an iterative development strategy is time-consuming, error-prone and boring. Therefore, we built a component that is responsible for loading the data and, if necessary, altering the table. Most software we built for this project is written in Python and C#.net. Only one project is in Java (PDF Parser). At forehand we know what data this software produces, thus in this case we choose not built this functionality for Java. The functionality built for Python is a module with a function that accepts a list of dictionaries, the name of the schema and the name of the table. The dictionaries does not have to be in the same structure. The procedure to load the data is as follows:

- Synchronizing the dictionaries. This means, all dictionaries contain the same key. If a key is available in one dictionary but not in another, it will add that key (with an null value).
- It creates a separate dictionary with the maximum size for each attribute. This information is needed to determine if a field in the database should be adjusted to another size. For ease of development, we only use character fields for ingesting data. In other layers of the solution this will be converted to correct data types and indexed for performance.

¹<https://www.postgresql.org/>

²<https://aws.amazon.com/rds/postgresql/>

³<https://cloud.google.com/sql/docs/postgres>

- If the table does not exist, it will create the schema (if necessary) and table.
- Otherwise, it checks for columns that are presented in the dictionaries but not in the table. If these are found, it will alter the table and add the column. If the column already exists, it will check if the new data can be inserted in the column, given its size. If not, the column will be altered to the new maximum size.
- Finally the data will be inserted using a binary insert. A binary insert is faster for inserting many rows compared with 'insert into'-statements.

6.1.2. TRANSFORMATIONS

Before the data can be used for analysis, it needs to be processed by cleaning and transforming procedures. This can be done by SQL-statements (Structured Query Language, language to interact with relational databases). The processes of cleaning, transformation and integration are called dataflows.

Requirements for our solution is transparency of the dataflow (how the tables and views are related) and testability. The tool we use is dbt⁴. The benefits of using this tool are:

- *Documentation*; the tool is able to create a website with documentation about the data lineage (what tables and views are input for a certain table) and description of the tables and columns.
- *Test framework*; tests can be added. An example of a test can be validating if the number of rows in the source table is the same as the number of rows in the target. This is useful for queries involving joins.
- *SQL dialect*; most databases have their 'features' (e.g. functions) on top of the standard ANSI SQL. With this tool you can write the SQL in the dialect of the database. This allows users to make full use of the database engine.

We did not encounter a framework or tool that has the same benefits. Also, dbt is open source, so functionality can be added if needed. Figure 6.2 (page 28) is a screenshot of the documentation functionality of dbt. On the left the database is shown (thesis) with its schemas (e.g. lncs_front_matter) and models (in this case dbt term for views and tables). In the middle a description of the table and columns (with datatype and description) are shown. On the right the lineage of this table is given. This shows which tables or views are input for the selected table, and where this table is used.

Now we discussed storage, the data needs to be acquired. The remainder of this chapter is about acquisition data of the web and from PDF documents.

6.2. WEB SCRAPING

Web scraping is a process consisting of two main steps: first acquiring the raw data (HTML page), second interpreting the raw data and transforming this in to information.

6.2.1. ACQUIRING WEB DATA

There are two main methods for acquiring data from the web. The first is to consider the website as an API and acquire the data via a GET request. Many software libraries exist for

⁴<https://www.getdbt.com/>

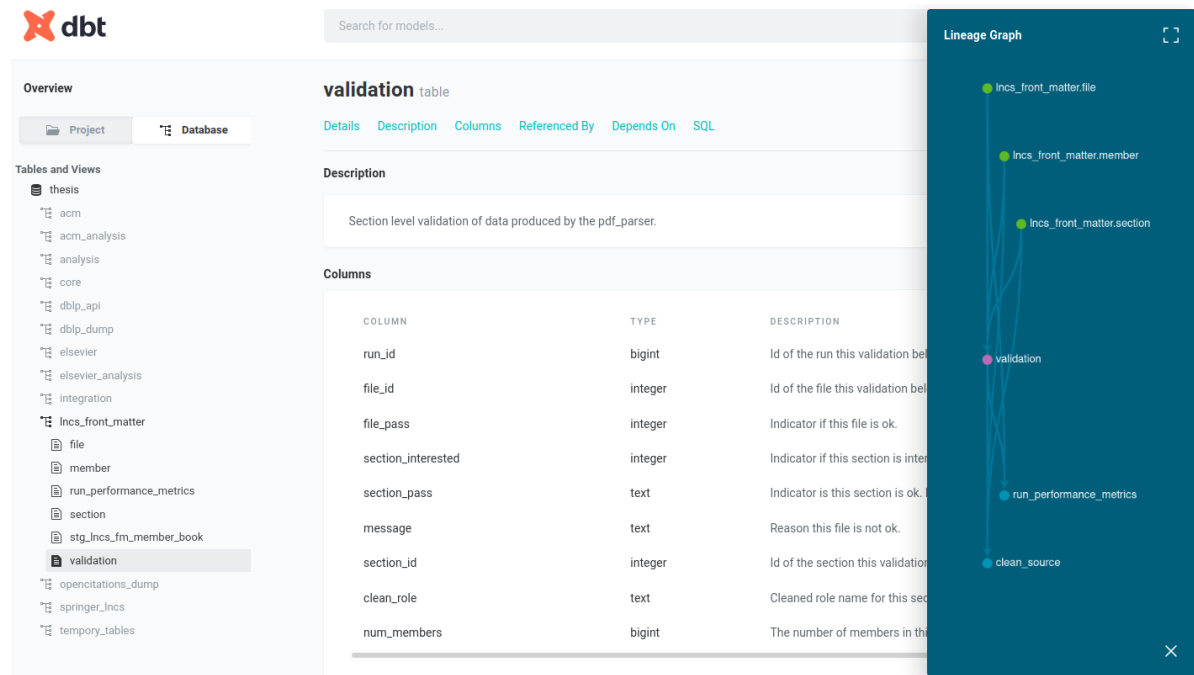


Figure 6.2: Screenshot dbt documentation functionality

doing this (e.g. Python requests⁵). The second is to control a web browser. Selenium is an example of such component that can control a browser and perform actions on behalf of a user. It is used for web testing, but the HTML of the resulting (e.g. generated) page can also be gathered. Choosing between these two options relies on the structure of the website. If the website contains plain HTML, using a request library would be preferred because of performance. A web browser control library is slower because:

- A web browser needs to be started;
- After every action performed, the tool need to wait until the page is loaded.

However, if a website is using asynchronous JavaScript calls to get the necessary data from the server (a dynamic web page), a web browser control library is a better option. Because it controls the browser, the actual JavaScript is being run and the content is delivered to the browser. If such a website is scraped by a request library, we only get the reference to the JavaScript instead of the content.

6.2.2. PARSING HTML

The data returned from scraping pages is formatted in HTML (HyperText Markup Language). For working with HTML, multiple libraries exists, e.g. BeautifulSoup⁶ for Python which can parse XML and HTML. However, the process of interpreting the HTML to extract information from the pages differs for every source; this cannot be generalised. The risk this approach contains, is that software built for this project may work at the time of writing, but fails if web pages changes.

⁵<https://docs.python-requests.org/en/latest/>

⁶<https://pypi.org/project/beautifulsoup4/>

6.2.3. IDENTITY HIDING

During the research preparation phase of this project, our requests to Science Direct started being blocked; our IP-address became blacklisted because of the amount of traffic. To work around this problem, two options exist:

- The first option is the use of a Virtual Private Network. Some free VPN providers exist and scraping sites using a VPN hides the IP for the website; the website only got the IP from the VPN server. The drawback is that the VPN provider may get blocked.
- The second option is using the TOR (The Onion Router) project as a proxy. TOR is a network of servers that hides the identity from the requester and encrypts the connection.

Because of the risk VPN servers got blocked, our first attempt was using a TOR proxy. Unfortunately, using the TOR network we experienced a lot of CAPTCHAs⁷ which made automating the scrape activities cumbersome. Therefore, the VPN approach is taken for this research project.

6.3. PDF SCRAPING

Some data of the publication process is publicly available, but scattered across websites (hence the web scraping) and other mediatypes meant for people to read and understand, but not machine-readable and therefore integrated in publication data sets. An example of such mediatype is the PDF (Portable Document Format) file. The main purpose of this file type is described by Adobe as to give “*people an easy, reliable way to present and exchange documents - regardless of the software, hardware, or operating systems being used by anyone who views the document*”⁸. In other words; focused on layout and intended for humans to read; not for computers.

Considering the amount of information stored in PDF documents (e.g. PC Members, Editorial boards) it is an unbearable task to copy and paste the data manually from these PDF's in a more structured format. Therefore, to be able to acquire the data inside these PDF's, we need to convert the PDF's to a structured or semi-structured dataset.

As stated in **Related work**, considering the error-rate of various tooling, pdftotext, pdftohtml, pdftoxml, pdfbox and parscit are considered the top 5 tools [BK17]. We experimented with some of the proposed methods, along with a visualisation method. Hereby we neglect the parscit tool, because this is specially for parsing citations in scientific papers. The following options are considered:

Option 1: Convert to HTML There are multiple tools available to convert PDF to HTML format. Further processing can be applied by using HTML libraries (see **Parsing HTML**). Unfortunately, the resulting HTML pages are complex to interpret because the content of a tag can contain only one letter. We need to glue these tags together, with taking the location of the tag in consideration (how much space is between two letters, is that a space, is that a new column, or an indent in an existing column etc.). We did an experiment to process PDF's this way, but the resulting solution to process one PDF was not able to process another one. The effort to abstract this functionality

⁷A CAPTCHA, Completely Automated Public Turing test to tell Computers and Humans Apart, is a 'gate keeping' mechanism to only allow 'real humans' to enter the site.

⁸<https://www.adobe.com/acrobat/about-adobe-pdf.html>

and make this a generic solution which can be extended, was very high in comparison with the result we achieved (high effort, low result).

Option 2: Convert to plain text Almost all libraries to read a PDF file support the option to read the file as plain text. Drawback is that the lay-out is gone. This lay-out is necessary to get some structure (and therefore ‘meaning’ of the text) from the document. Without this lay-out we are unable to identify text as a section header, or identify columns in the document.

Option 3: Convert to image and extract text We can use machine learning libraries to get the text from an image. We tried this with Tesseract⁹, but too much information was lost to get the structure of the document. The same issues applied as with the ‘convert to plain text’ option.

Option 4: Use low level library In another experiment we tried if we were able to process PDF documents with a more low level library. Using such library gives us more control of the process. First small experiments with PDFBox¹⁰ were unsuccessful; these also outputs the text without layout. However, because the low-level control we have by using this library, we can adjust this to our need.

We choose to proceed with option 4, PDFBox, because of the extensibility and control we have and, as mentioned before, other solutions were unable to keep the necessary structure (option 2 and 3) or needed too many adjustments to make it work with more and other formatted documents (option 1).

As a first step of generic interpretation we need to get the raw text from the PDF. PDFBox comes with a parser out-of-the-box that gives the PDF back as plain-text. The problem is within this text all spaces between words are converted to one space. As mentioned before in option 2, this is not sufficient for documents containing information about the publication process; we need to be able to identify columns, so we need to keep all spaces.

Luckily, we were not the only one with this problem. An open source parser for PDFBox is made available on GitHub¹¹ that keeps the layout as is. However, this extension still did not have all properties of the text we need to get the structure of a file; it only keeps the spaces between words. Because this parser is open source, we are able to adapt this parser to our needs; we can add additional properties to the text objects we read from the PDF. Besides the text with all the spaces in between (to preserve the layout) we get by using this extension, we also added the size of the font and the layout (bold identification).

The end result of this phase is that we have the PDF file as textobjects (lines of text with properties, e.g. size) which we can use to transform the raw text in information. However, this interpretation part is specific to the kind of information we want to extract from the document.

⁹<https://github.com/tesseract-ocr/tesseract>

¹⁰<https://pdfbox.apache.org/>

¹¹<https://github.com/JonathanLink/PDFLayoutTextStripper>

7

CASE STUDY 1: PROGRAM COMMITTEE MEMBERS

This chapter focuses on the Program Committee Members as group. We will start with a motivation why this group is interesting. After this motivation we will investigate how we can put the analysis dataset together. We continue with performing some analysis and validation which gives (in combination with the other case studies) answer to the main question of this study.

7.1. MOTIVATION

The most important motivation to investigate this group, is that this group is in a position with great power. In the end, these people decide which articles (and which authors) get published. Implicit, they decide whose career is going to thrive. This power becomes evident in an experiment conducted on an Artificial Intelligence conference: the NIPS experiment.

The NIPS experiment. For scientists getting a paper in a conference is important for their career. Unfortunately, the NIPS experiments makes clear that reviewing papers for a conference is not a deterministic process¹. If a paper gets approved depends on the composition of the responsible committee. This means that the committee judging papers for conferences has the power to determine whose paper is being accepted and therefore whose career is going to thrive.

PC Members are in a position in which they can exploit this power. Members with high impact are members on high impact conferences. For this reason, we focus on top conferences. The goal with this case study is to see if we can identify outliers within a group of PC members. An interesting case related to this group is described in the next section.

7.1.1. INTERESTING CASE: WORK PC MEMBER IS BEING CITED

When an author becomes a specialist of a certain subject, it is not uncommon that this person becomes member of the editorial board or program committee of a venue that handles

¹<http://blog.mrtz.org/2014/12/15/the-nips-experiment.html>

this specific subject. New publications regarding this subject will likely be published in this venue. It is not strange that this publication cites work of the person-of-interest, if he is indeed a specialist. In Figure 7.1 we present an instance model of this situation. In this model, pers1 represents the person-of-interest. We notice the relation of this person to the venue (v1) as an editor and to previous work (p2) as an author. A new publication (p1) has a citation to the previous work of the person-of-interest (c1).

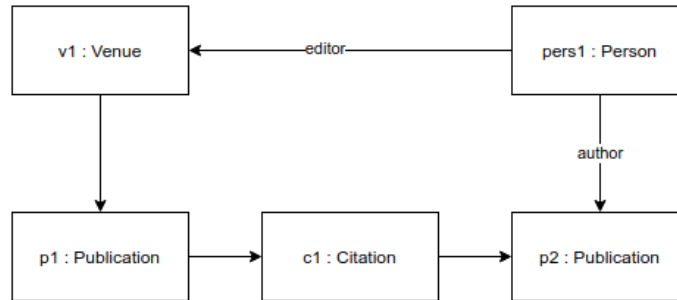


Figure 7.1: Instance model of citation of work of member from editorial board

Attack. When this situation is being enforced (malicious intent), it becomes attack (e.g. the member says: "cite me, or your work will not be published"). The person of interest benefits from this situation; he will be cited more, which impacts his metrics (e.g. H-Index). The question is how this enforcement impacts the publication model so we are able to detect this.

Model impact. The impact this attack has on the publication model is on the objects Venue, Person (with the role editor) and the relationship between. This relationship is the fact that a person works for a venue. In this relationship, citations occur; while a person works for a venue his work is being cited. In case of an attack; we see more citations than usual. The metric we need from this relationship is the number of citations: *number_of_cites*. But we need to take the time into account; if a person works for a venue for a long time (multiple incollection or inproceedings are published), it is to be expected the number of cites while being a member is high. Because of this; we also need the number of incollection and inproceedings: *number_of_issues*. These metrics can be used for comparison by calculating the 'normal' and the deviation of all members. This 'normal' can be calculated over the venue the editor is working for, or all venues. This comparison makes it possible to identify interesting editors in comparison with his peers.

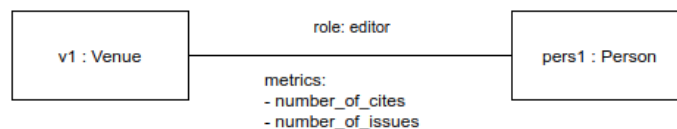


Figure 7.2: Metrics needed for detection on publication model

Benign alternatives. The described impact on the publication model can also have other benign reason to occur. The detection strategy is not foolproof. We need to take into account that a person is such an excellent contributor, it is logical he is being cited far more than his peers.

7.2. IMPLEMENTATION

In this section we elaborate on the implementation of the flow to acquire and integrate the data.

7.2.1. FUNCTIONAL OVERVIEW

The focus of this case study are people on a position with influence. People with influence can be found at top conferences and journals. Therefore, the primary selection criteria are top conferences and journals. These conferences are the input for the proceedings we need to collect. From these proceedings we need to get the articles published in the proceedings and the PC Members involved. From the articles, we need to get the citations. In Figure 7.3 we draw this overview. The arrows mean that, in case of the left arrow, the top conferences are input to get the proceedings (meaning only proceedings of the top conferences).

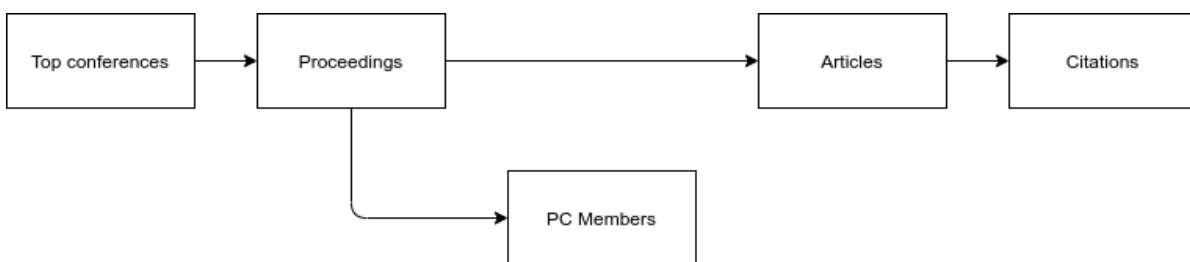


Figure 7.3: Functional overview

From this functional overview of the situation we zoom into the implementation for acquisition and interpretation.

A word about notation. In this (and the other case studies') implementation chapter we draw elements to represent applications and its inputs and outputs. See Figure 7.4 for the notations used. The left box represents an application. The text in this box means that the application can be found in the final repository² in the directory 'solution/subdir1/subdir2'. The name of the (in this case Python script) is called 'app.py'. In case of a .Net application, the text is '.net' and next line the directory of the application. The parallelogram in the middle represents a relational storage in the database (table or view). The name on top is the schema name, the name under the schema name is the name of the table or view. On the right we draw a cube which represents disk storage. The text between brackets is the type of files stored.

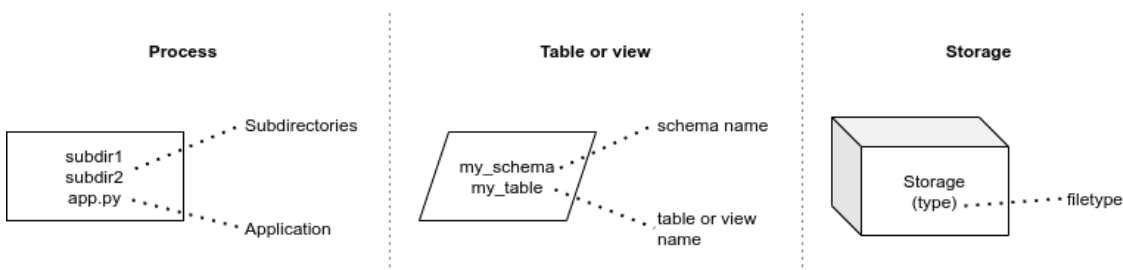


Figure 7.4: Notation usage (left: application, middle: table or view, right: storage)

²On GitHub <https://github.com/woudw/IM9906>

7.2.2. TOP CONFERENCES

The focus of this case study is people on a position with influence. People with influence can be found at top conferences and journals. Therefore, the primary selection criteria are top conferences and journals. For our study, which focuses on computer science, we get the top conferences and journals from The Computing Research and Education Association of Australasia (core)³. Core scores conferences with the following ranking⁴:

- A* Flagship conference
- A Excellent conference
- B Good to very good conference
- C Other ranked conferences with minimum standards

Besides these four, they also have 'Australasian', 'Unranked', 'National', and 'Regional', which is not of interest for our purpose. Based on their ranking, we select the conferences for our research with a ranking of B or higher. The CORE website provides an export functionality for acquiring data. However, for integration purposes, this is not sufficient; the export misses the link to the DBLP conference, which is available on their website. For that reason, we built a web scraper.

CORE SCRAPER

The scraper should be able to get the necessary information for our case study. This tool does not depend on any input; it scrapes all data of the website. In Figure 7.5 we position this tool as implementation to fulfill the functionality to get the top conferences. As shown, this tool has two output tables: one for conferences and one for journals.

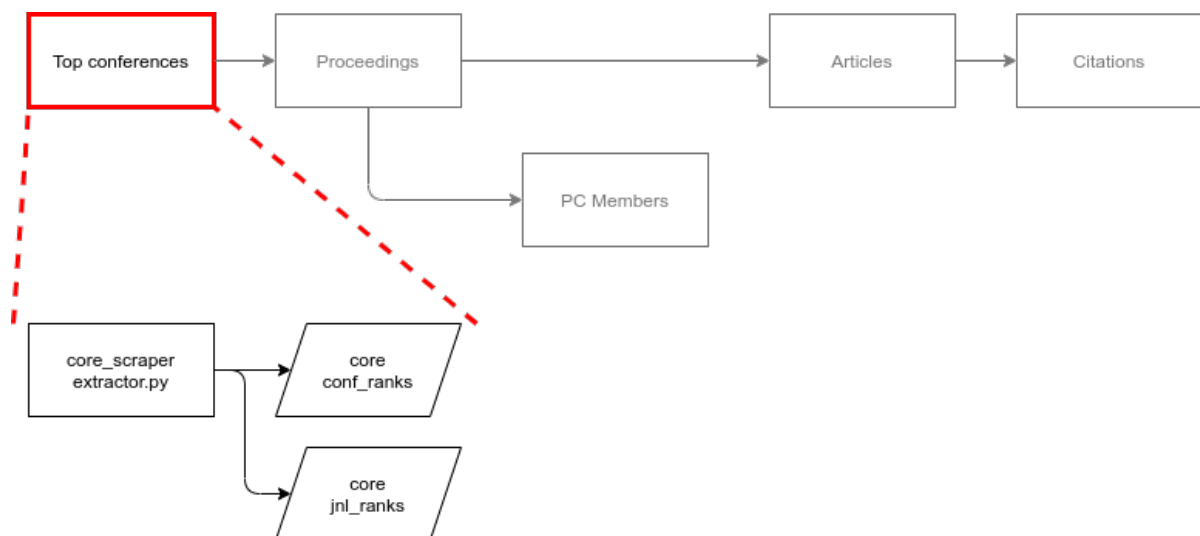


Figure 7.5: Top conferences implementation

This tool does not store the data to disk before interpretation. The reason is that the data is represented in an HTML table (which is highly structured) and therefore can be directly inserted in the database. The output of this scraper is not sufficient to acquire the proceeding information. Two steps need to be added:

³<https://www.core.edu.au/>

⁴<https://www.core.edu.au/conference-portal>

- The dataset misses 17 DBLP links, we add these by hand.
- We need to filter the conferences on the ranking.

These steps are implemented in dbt (Section 6.1.2).

7.2.3. PROCEEDINGS

With the selection criteria in place, we get the proceedings of the conferences from DBLP. DBLP provides the option to download the complete data dump from their website (described in Chapter 5). Unfortunately, conferences are not part of this dump. But, with the API of DBLP we are able to get the proceedings based on the searchterm from the DBLP url of CORE. Therefore, we built a tool to load this data from the API in the database.

DBLP API EXTRACTOR

In Figure 7.6 we position this API extractor in the functional landscape.

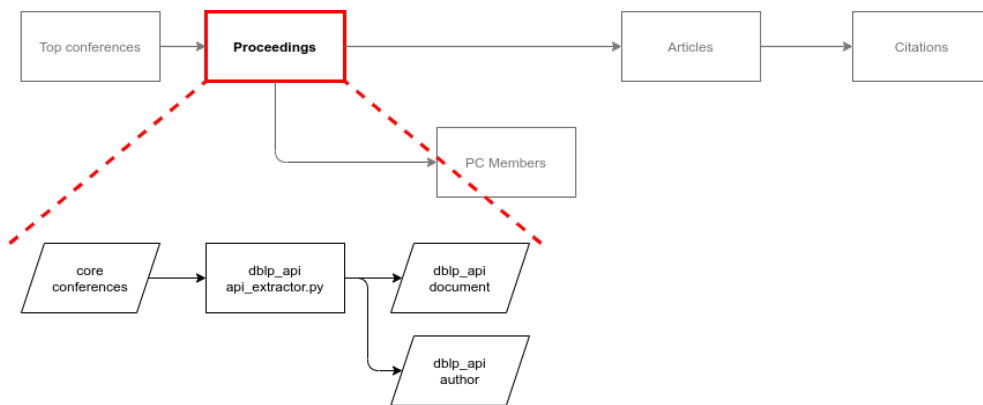


Figure 7.6: Proceedings implementation

As shown in Figure 7.6 the input is the table `core.conferences`. This table is the output of the core scraper with the additional steps. For every DBLP key in this conference set, we request all proceedings.

The output of this application are two tables: document and author. We focus on document, which are the actual proceedings (author are the editors). In Figure 7.7 we show the publishers with more than 1,000 proceedings.

Publisher	#proceedings
Springer	4,751
ACM	4,115
IEEE Computer Society	2,422
IEEE	1,210

Figure 7.7: Publishers with >1,000 proceedings

For this case study we want to use the biggest source, which is Springer. Springer publishes their proceedings under the Lecture Notes of Computer Science (LNCS).

The resulting set contains 4,467 unique proceedings (the DBLP API search function sometimes returns double results).

On the site of a LNCS proceeding, a front matter document is available for download. A front matter document contains information about the proceeding, including people involved in that specific conference; e.g. steering committee, reviewers and, important for our case, the PC members.

The data acquired from the DBLP API contains the DOI link to the proceeding on LNCS (for 1 document this misses, we added it manually). With this anchor we can start acquiring the necessary data, including the download link to the front matter document.

7.2.4. PROCEEDING INFORMATION, ARTICLES AND AUTHORS

To get the data from LNCS we built a web scraper. In Figure 7.8 we position this scraper in the functional landscape.

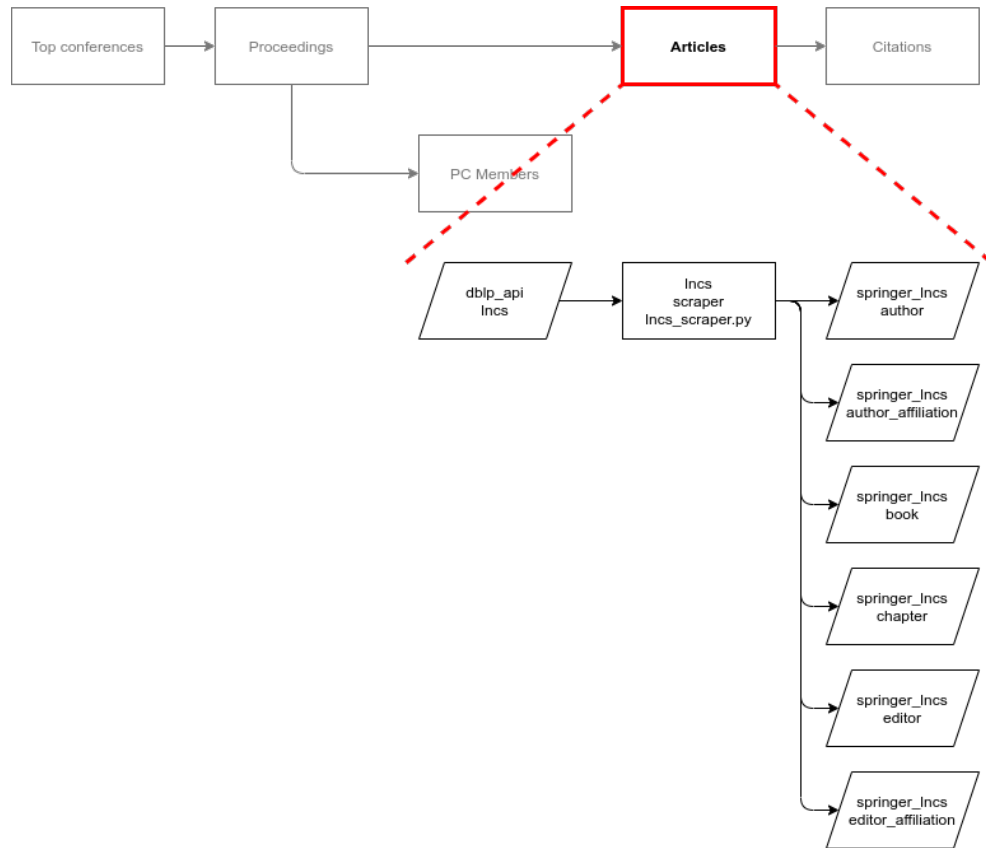


Figure 7.8: LNCS web scraper implementation

The input for this web scraper are the DOI links from the DBLP API. The output are six tables: books (proceeding), chapters (article), editors and authors and their affiliation. See Figure 7.9 (page 37) for a graphic representation of the entities and their relations. The relationships are 1 to many. The book table is the actual proceeding. A proceeding is edited by an editor. A proceeding contains multiple chapters (LNCS terminology for article). A chapter is written by one or multiple authors. The authors and editors may have one or many affiliations (e.g. universities, institutes, companies).

The output of this application is not stored to disk, but directly to the database. The reason is that we need to interpret the proceeding page to get the article links; and while we are interpreting the page, we interpret the other information as well. Also, the data in the HTML page is mostly key-value based. As an example we show a piece of HTML of a proceeding page in Listing 7.1. This listing is a list (tag) of items (tag). Every item has a span with classname ending in 'title' (the key) and a span with a classname ending in 'value'. Because of this standardization used on the page, we can loop through the items

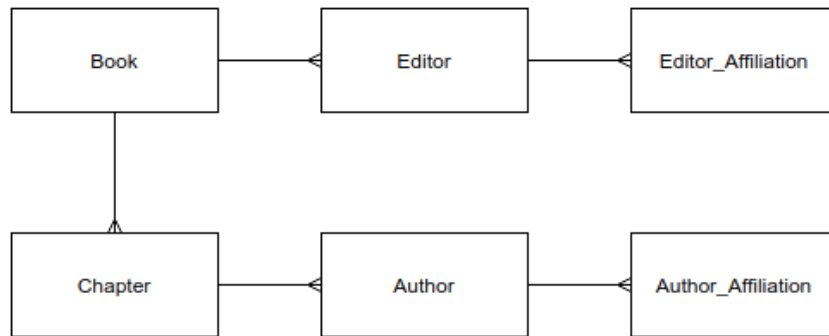


Figure 7.9: Datamodel of the output of the LNCS scraper

and scrape all data. Combined with the automatic adjustment of the database table (see Section 6.1.1) we can add all these key-value pairs to a dictionary and store them in the database.

Listing 7.1: Example HTML LNCS proceeding

```

1 <ul class="bibliographic-information__list--inline">
2   <li class="bibliographic-information__item">
3     <span class="bibliographic-information__title">Book Title</span>
4     <span
5       class="bibliographic-information__value u-overflow-wrap"
6       id="title">Mastering Scale and Complexity in
7       Software Reuse</span>
8   </li>
9   <li class="bibliographic-information__item">
10    <span class="bibliographic-information__title">Book Subtitle</span>
11    <span
12      class="bibliographic-information__value u-overflow-wrap"
13      id="sub-title">16th International Conference on
14      Software Reuse, ICSR 2017, Salvador, Brazil, May
15      29-31, 2017, Proceedings</span>
16  </li>
17 </ul>

```

The reason we also acquire the chapters is to get the relation between a book and the articles it contains. To keep our main goal clear: we need to get the people involved, so while we are scraping the articles, we can scrape the authors (and their affiliations) just as well. The most important entities we gather from LNCS are the books (proceedings), chapters (articles) and authors. The editors and their affiliation are not important for this case study. All chapters we scraped contain a DOI as attribute. By scraping the books we also get the link to the front matter PDF which can be used to download the front matter, which contains the PC Members.

7.2.5. PC MEMBERS

Acquiring the PC Members is a two-step process. First we need to download the PDF documents. The input are the URL's to the front matter. These are available as attribute of the book (proceeding) object from the previous step. For this we created a download tool which stores the data on disk. As name of the stored PDF we use the DBLP name of the

proceeding so we are able to relate the PDF document to a proceeding. After downloading the front matter documents, we need to parse the documents to get the required information. An overview how these two applications fit in the landscape is shown in Figure 7.10. After the figure, we will dive in the details of the PDF parser and explain the tables shown as output (the red lines are for visual distinction, they have no functional meaning).

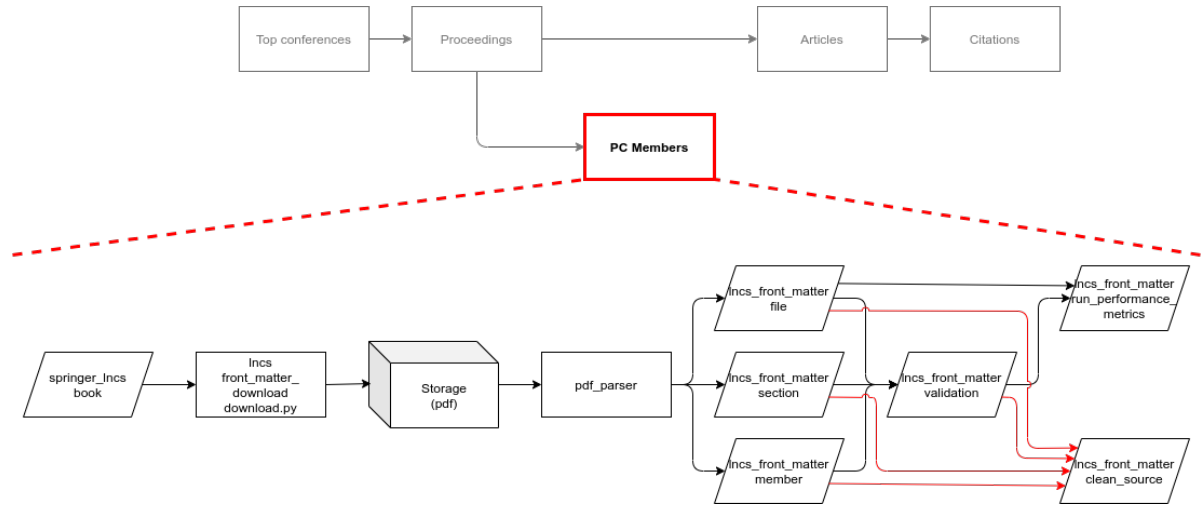


Figure 7.10: Acquisition of PC Members implementation

ACQUISITION OF INFORMATION FROM THE FRONT MATTER DOCUMENTS

In Section 6.3 we described the generic parsing of PDF documents to textobjects which can be used for interpretation; converting content to information. Currently we have 4,370 Front Matter documents. To get the information from these documents we built a prototype of a parser.

Building a document tree. By using the properties `textsize` and `bold` properties of the textobjects we are able to identify headers in the document which we need to build an hierarchical structure of the document. In most front matter documents there is an organization header, with subheaders containing the role of the people. The `textsize` is also used to identify content; these are textobjects with the same size as the size which occurs most in the document. Sometimes documents contain a footer or header with a smaller `textsize` as the content. By only using textobjects with the size as the content, we ignore these footers and headers.

To be able to work with the document structure, we parse this hierarchical structure in a tree datastructure. The objects we store in this tree are Sections. These contain the textlines and the header as title of the section. In Figure 7.11 an organization part of a front matter document is shown. The format of this part makes clear that we need properties as text size and bold identification to identify headers.

In Listing 7.2 a textual representation of a part of Figure 7.11 is shown. The most important for our purpose is the section called 'Organization' with the underlying sections 'Steering Committee', 'General Chairs', 'Program Chair' and 'Workshop Chairs', as these are the roles of people involved in this proceeding. From the entire document tree, we can easily find the section that contains the organisation and all sections underneath.

Organization	
Steering Committee	
Katsushi Ikeuchi	The University of Tokyo, Japan
Yasushi Yagi	Osaka University, Japan
Tieniu Tan	The National Laboratory of Pattern Recognition, China
General Chairs	
In So Kweon	KAIST, Korea
Chilwoo Lee	Chonnam National University, Korea
Akihiro Sugimoto	National Institute of Informatics, Japan
Program Chairs	
Kyoung Mu Lee	Seoul National University, Korea
Yasuyuki Matsushita	Microsoft Research Asia, China
Jim Rehg	Georgia Institute of Technology, USA
Zhanyi Hu	Chinese Academy of Science, China
Workshop Chairs	
Jongil Park	Hanyang University, Korea
Junmo Kim	KAIST, Korea
Hideo Saito	Keio University, Japan
Yanxi Liu	The Pennsylvania State University, USA
Ming-Hsuan Yang	University of California at Merced, USA

Figure 7.11: Organisation part in a front matter

```

...
- - - {"section":{"title":"Organization", "# lines":8}}
- - - - {"section":{"title":"Steering Committee", "# lines":7}}
- - - - {"section":{"title":"General Chairs", "# lines":7}}
- - - - {"section":{"title":"Program Chairs", "# lines":7}}
- - - - {"section":{"title":"Workshop Chairs", "# lines":8}}

```

Listing 7.2: Representation of the organisation part

Processing a section. A section is a textpart of the document. In Figure 7.12 an example is shown. In this example, the role of the member is the title (Steering Committee). The text in a section with people involved in the proceeding, has a grid structure. A visual representation of a grid laid on top of Figure 7.12 is shown in Figure 7.13.

Steering Committee

Katsushi Ikeuchi	The University of Tokyo, Japan
Yasushi Yagi	Osaka University, Japan
Tieniu Tan	The National Laboratory of Pattern Recognition, China

Figure 7.12: Section in the Front Matter

Katsushi Ikeuchi	The University of Tokyo, Japan
Yasushi Yagi	Osaka University, Japan
Tieniu Tan	The National Laboratory of Pattern Recognition, China

Figure 7.13: Section in the Front Matter

Because of this structure, we use a grid as data structure to put the data from the raw lines into. In Listing 7.3 we show the parsed section in a grid representation from the logs files. The challenge in this part is to merge lines into one line. This process is done by dedicated section-to-grid object. Separating this process from the process of building a tree, improves testability.

```
+-----+-----+
| Katsushi Ikeuchi | The University of Tokyo, Japan |
+-----+-----+
| Yasushi Yagi     | Osaka University, Japan |
+-----+-----+
| Tieniu Tan       | The National Laboratory of Pattern Recognition, China |
+-----+-----+
```

Listing 7.3: Representation of the grid (captured from the application logs)

This grid structure gives us some possibilities, e.g. to get statistics of a column, which we are going to need to choose the correct parser to transform this text into information.

Getting information from a grid. To get information from a grid we need to parse the text in the grid. For this we define a parser interface. The implementation of this interface depends on the structure of the grid (e.g. the number of columns) and the content (e.g. is it an affiliation or a name).

Determining which implementation of the parser is needed, is done by a `GridParserFactory` which uses the statistics and format of a grid. The properties collected from the grid to choose the correct parses are:

- *Number of textparts*: The total number of cells that are filled in the grid.
- *Number of columns*: The number of columns that the grid contains. Most sections contain two columns, but this is certainly not the case for all sections.
- *Affiliation ratio*: The ratio of textparts that contain keywords for an affiliation (e.g. "universi"). This is calculated for every column and for odd, even and all rows.

- *Comma ratio*: Same as affiliation ratio, but for commas. Can also be used to identify affiliations. However, names also contain commas if the format is lastname, first-name.

The affiliation ratio and comma ratio is calculated for odd, even and all rows. The reason is that sometimes the affiliation is presented under the name of the member. In Listing 7.4 we show some statistics for identifying affiliations.

```
numberOfColumns:      2
affiliationRatios:
  {"columnNumber":0,"rows":"ALL","ratio":0.0}
  {"columnNumber":0,"rows":"ODD","ratio":0.0}
  {"columnNumber":0,"rows":"EVEN","ratio":0.0}
  {"columnNumber":1,"rows":"ALL","ratio":0.6666666666666666}
  {"columnNumber":1,"rows":"ODD","ratio":0.5}
  {"columnNumber":1,"rows":"EVEN","ratio":1.0}
```

Listing 7.4: Some derived information from the grid

With this information the GridParserFactory can choose the appropriate implementation of the parser to interpret the data correctly and convert the grid into information (members and affiliations). For example, Listing 7.4 shows us that the ratio of affiliation keywords in the second column (0-based) is above a certain threshold and the section consists of two columns, the software chooses a Two_Name_Affiliation parser which means two columns, first column is a name, second column is the affiliation. The resulting objects are shown in Listing 7.5.

```
{
  "name":"Katsushi Ikeuchi",
  "affiliation":"The University of Tokyo, Japan",
  "role":"Steering Committee"
}
{
  "name":"Yasushi Yagi",
  "affiliation":"Osaka University, Japan",
  "role":"Steering Committee"
}
{
  "name":"Tieniu Tan",
  "affiliation":"The National Laboratory of Pattern Recognition, China",
  "role":"Steering Committee"
}
```

Listing 7.5: Resulting information (from the logs, reformatted for readability)

OUTPUT

This process to parse PDF files to usable information creates the following data:

- *Information dataset*: The process outputs the members (name and affiliation), the role and the document this information was read from.
- *Metadata*: Besides this information, the statistics described in ‘[Getting information from a grid.](#)’ and the used parser implementation is also added. We use this to validate the results; we will come back to this in ‘[Validation](#)’.

- *Logging*: For every PDF the application parses, a log file is created. The purpose is error-detection and the possibility to quickly gather information to extend the software and create the tests; iterative logging-based development.

Iterative logging-based development process. For every PDF that the program processes, it keeps a separate logfile. This gives us the ability to investigate why the program was unable to process some files or sections. Missing parser for example can be detected using this mechanism in combination with the data in the section table in the database.

For every file the logfile shows the document tree (e.g. Listing 7.2). If sections are found, the logfile shows for every section:

- *Textlines*: All the lines that are in this section.
- *Grid*: The derived grid from the textlines (e.g. Listing 7.3).
- *Members*: The members that are extracted from the section.

Having this information gives us the possibility to investigate in detail why data can not be interpreted and easily build new functionality (e.g. new parser implementations) and add unit tests for this functionality.

VALIDATION

Currently 4,370 front matter documents are available. How are we able to tell something about the quality of the program without manually going through the results?

For every section we gather some metrics (e.g. number of lines read, number of lines merged, chosen parser). This data is stored in the output dataset (file, section, member) shown in Figure 7.10 on the right side. With these metrics, depending on the parser we can see if the number of extracted members is correct. This data is stored in the validation table. For example, the parser `Two_Role_NameAff` contains two columns: first with a name and second with the affiliation. We can measure if the number of resulting members is equal to the number of non-empty lines in the section minus the number of merged lines. If it is, we can use the extracted members with confidence; the data is read correctly.

Based on these 4 datasets (file, section, member and validation) we can create 2 tables: one table with performance statistics of the run (e.g. how many sections are corrected interpreted) and a dataset which contains 'cleaned' data. This cleaned data transforms the different roles to a standard set of roles and only contains validated data; this makes sure this use case only uses validated data. All these transformations are done using SQL (dbt, Section 6.1.2).

Current state. In Figure 7.14 we show a tree with the file counts. The input for this tree is the performance statistics table described previously. From the 4,370 files we were unable to process 1,237 files of which 1,119 because the software was unable to find the organization header. It may be useful for improvement of the software to tackle this issue first. Of the correct processed files, 2,223 contained interesting sections. These are section with the role 'program committee'. Among these interesting section, 2,711 across 2,055 files are validated with the above described method and can be used during analysis. This 'good' path is shown in red.

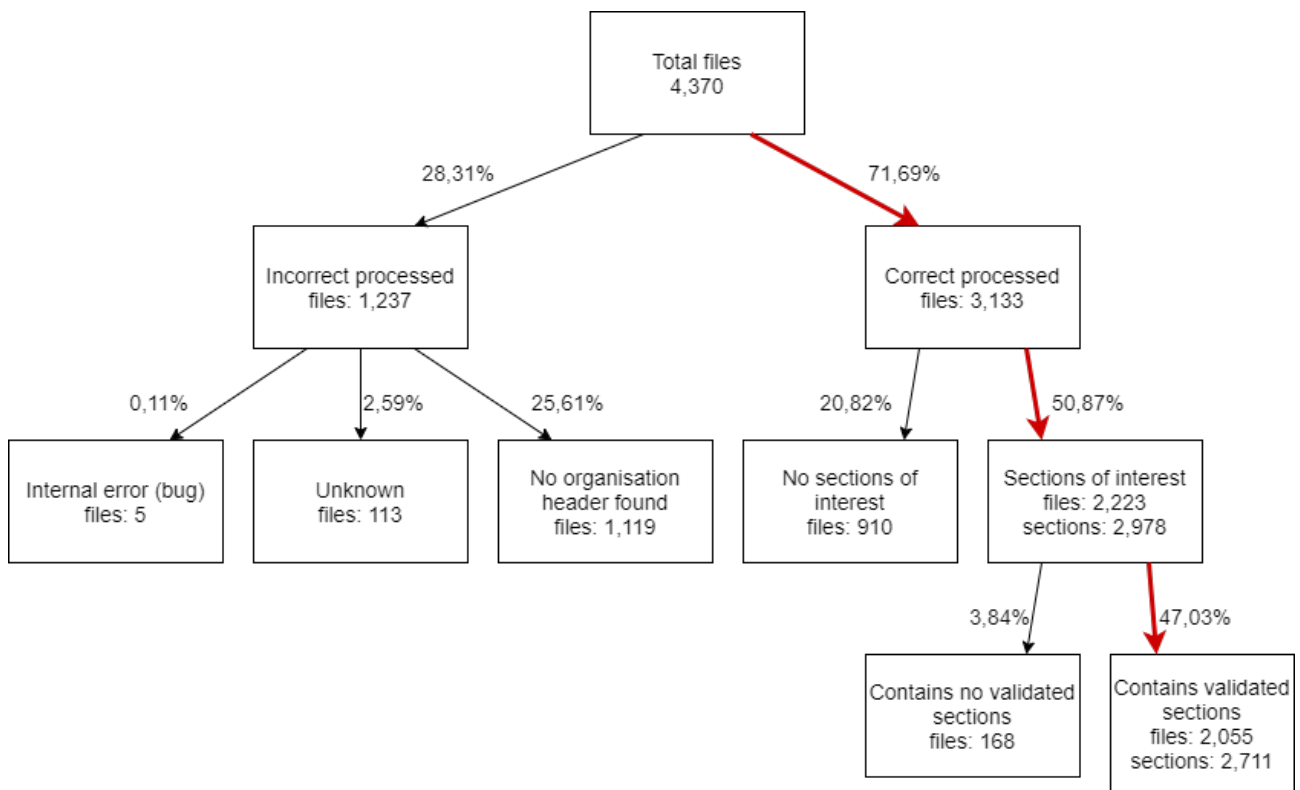


Figure 7.14: Current results

7.2.6. CITATIONS

Sources to acquire citation information are discussed in Section 5.2. All chapters (articles) we scraped from the website contain a DOI as attribute. Therefore, the source we use is OpenCitations (this source is DOI-based). The data of OpenCitations can be downloaded in CSV format or requested by an API. An experiment using the API resulted in a very slow process of acquiring the data. Therefore, we choose to download the data. This data is stored on disk storage. However, this dataset contains a lot of data separated over multiple files. Not all data is required; only citations of articles from the LNCS proceedings. To process the CSV files from OpenCitations for only the articles of LNCS, we need a software component. In Figure 7.15 (page 44) we place this application in the functional landscape. The input for this application are the stored CSV files and the chapter table acquired from scraping the LNCS website for the DOI attribute.

The tool iterates through the files (every line is one citation) and stores the citation if the ‘from’ DOI is in the set of articles. However, because of the amount of files and data, this tool will benefit from some performance improvements:

- *Parallel processing*: this tool is multi threaded and processes the amount of files in parallel same as the amount of CPU’s available.
- *Caching of the article DOI’s*: because the tool needs to check if the ‘from’ DOI is in the list of article DOI’s, we need to cache this list. We tried three options: a list of strings, an array of strings and a HashMap. The HashMap was significantly faster.

The output of this application is one table with citations from an article in the LNCS proceedings to other DOI’s.

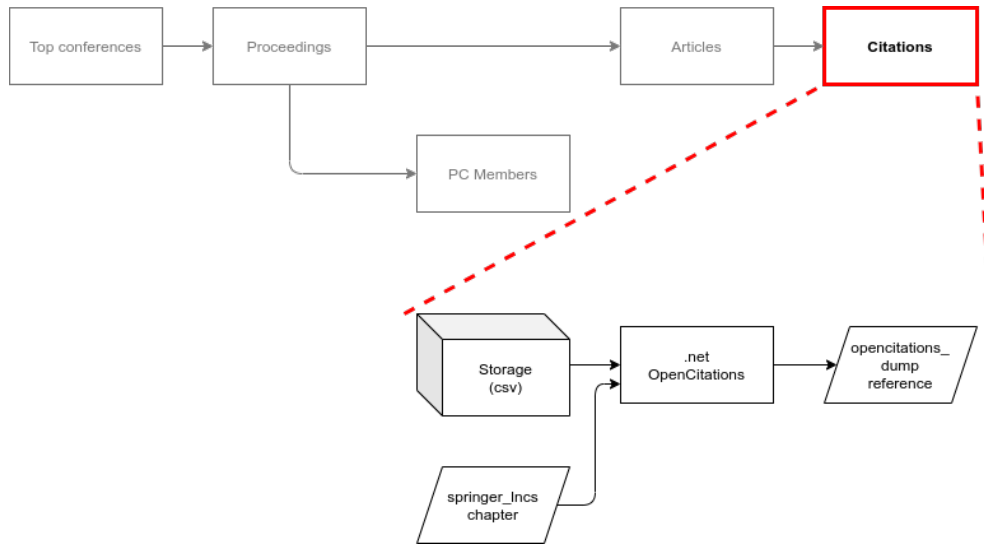


Figure 7.15: Acquisition of citations.

7.2.7. FORMALISING THE INTEGRATION DATASET

In this section we explain how we integrate the data we collected from the sources.

In the resulting dataset we combine the DBLP dump (see Section 5.2.1) with the data acquired from the previous section. In Figure 7.16 we show the sources and their relationships to create this dataset. Explanation will follow after the figure.

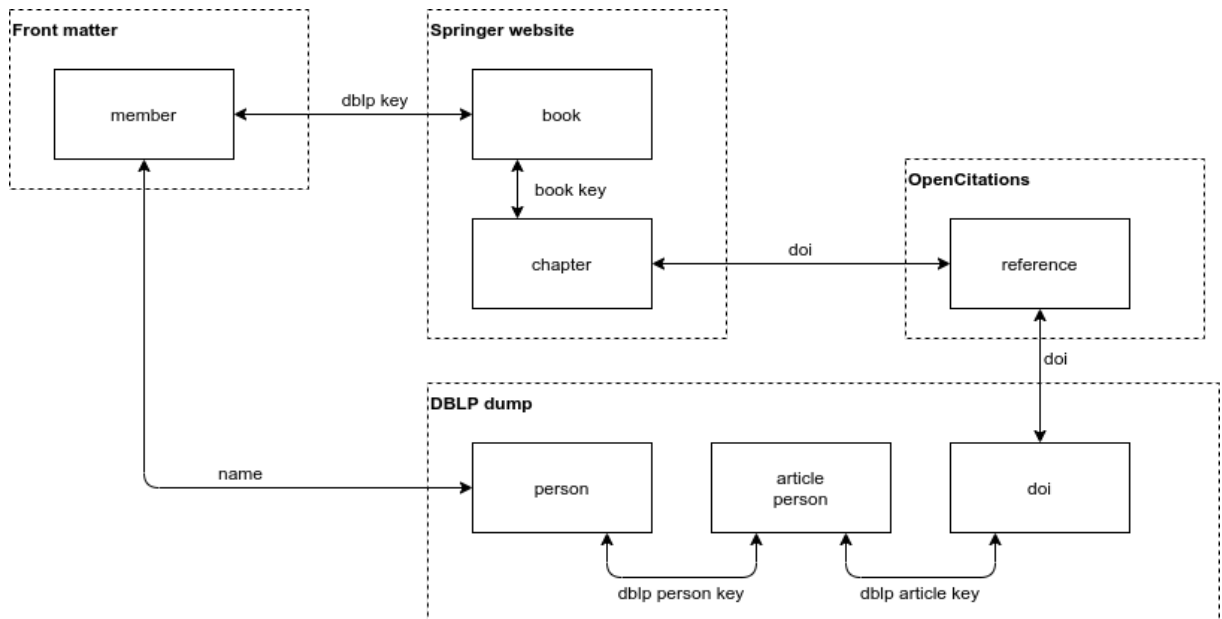


Figure 7.16: Formalising the citing dataset by joining sets based on keys

We will describe this set clockwise starting in the top left. The keys and foreign keys within a source (DBLP dump, Springer website) are not described here. The front matter *member* (upper left) is the result of applying a filter to use validated data. Because we use the *DBLP key* in the name of the PDF file, we can join this set with the proceeding (*book*) of the Springer website. The Springer website part (top middle) is described in Subsection 7.2.4. The interested entities for this set are the book and the *chapters* (articles) in this

proceeding. From the chapter we get the *reference* by the *DOI* attribute (chapter DOI to reference ‘from DOI’). From this reference set we get the referenced articles from the DBLP dump by using the ‘to DOI’ to the *DOI* in DBLP dump. Within the DBLP dump we have the names of the authors of this referenced article in the *person* set, which we can eventually use to go back to the *member* name.

Two choices are made by composing this set:

- The relationship on the *name* from *person* to *member* is optimistic; the join is on the raw name. However, because we use DBLP we match against possible multiple names of the author (see Section 5.3.2).
- Because we use OpenCitation, we only refer to articles in DBLP which contain a DOI. Another consequence is that we rely on the completeness of OpenCitation, but this is an issue with every source we use.

The implementation of the flow is added in Appendix A.

ADDED VALUE OF THESE DATASETS

By having these datasets we can calculate citation metrics for PC members and compare them with their peers, which may be in the same proceeding or across proceedings, depending on the scope of the peers.

7.3. ANALYSIS AND VALIDATION

In this paragraph we validate if acquiring the data with the focus on PC member citations actually help to identify cases which can be further manually investigated. From the defined dataset we have the for every member the *count the members’ work is cited in the proceeding he was a member of*. As analysis we focus on citations from Symposium on String Processing and Information Retrieval (SPIRE) as an example. The dataset contains observations of 483 PC members over 17 years (2003 - 2020). In Figure 7.17 we show the frequency of citations. We notice a large number of zero citations, which means that members during an activity (being PC member for a certain proceeding) are not being cited.

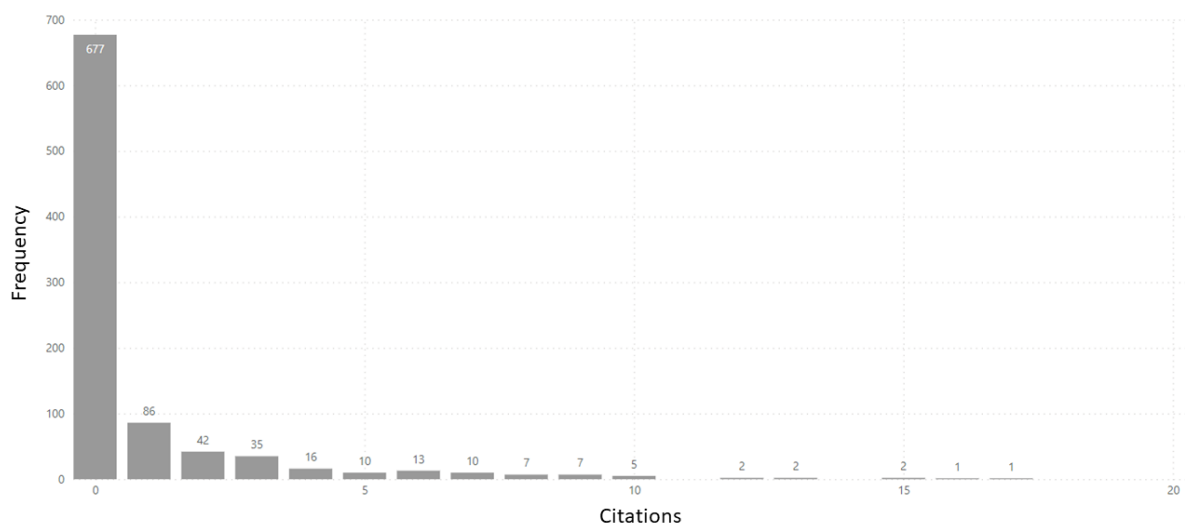


Figure 7.17: Frequency of citation count

In this phase we need to set the threshold to define an outlier. Eventually this should be a decision made by domain experts of the publication process. For our purpose and as an example analysis we decide to place a threshold at 10 citations; every activity with more than 10 citations is marked as an outlier.

In Figure 7.18 we plotted a few PC member activities. Every dot is a member for a certain year of the publication. The X-axis is ordered by member and year. E.g. we see that Member A has 3 citations for the 2018 proceeding. The Y-axis is the number of citations. The red dot in the top-right indicates that an activity is an outlier. By plotting the data this way, we can quickly see clusters of outliers, which may indicate members that continuously have more citations than their peers.

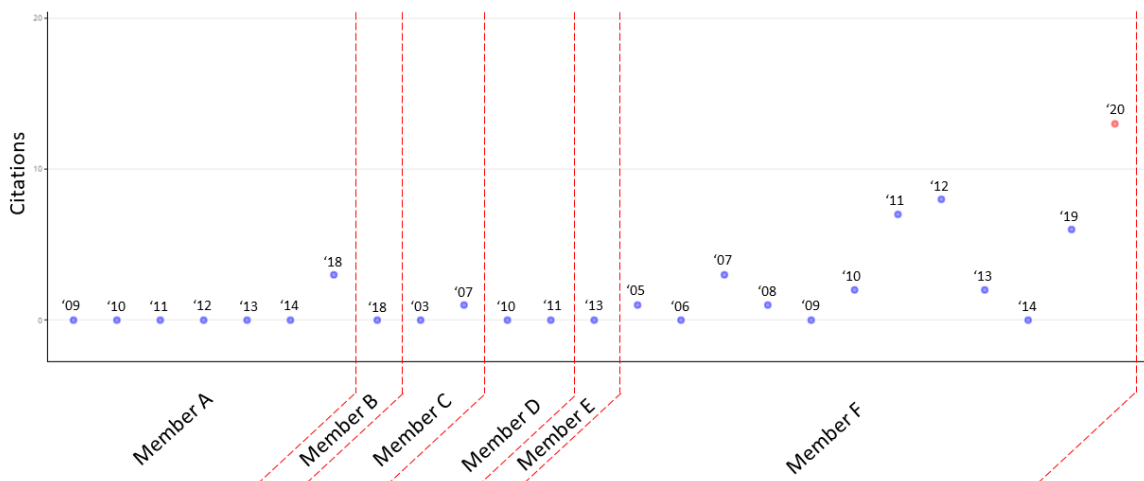


Figure 7.18: Example of citation data from Symposium on String Processing and Information Retrieval (SPIRE). (Red lines added for clarity)

In Figure 7.19 we plotted PC member activities for all PC members in SPIRE. The red dots are identified as outliers. One can clearly identify a cluster (circled by a red dotted-line) which is the same PC member with activities in multiple years (the name is known by the authors).

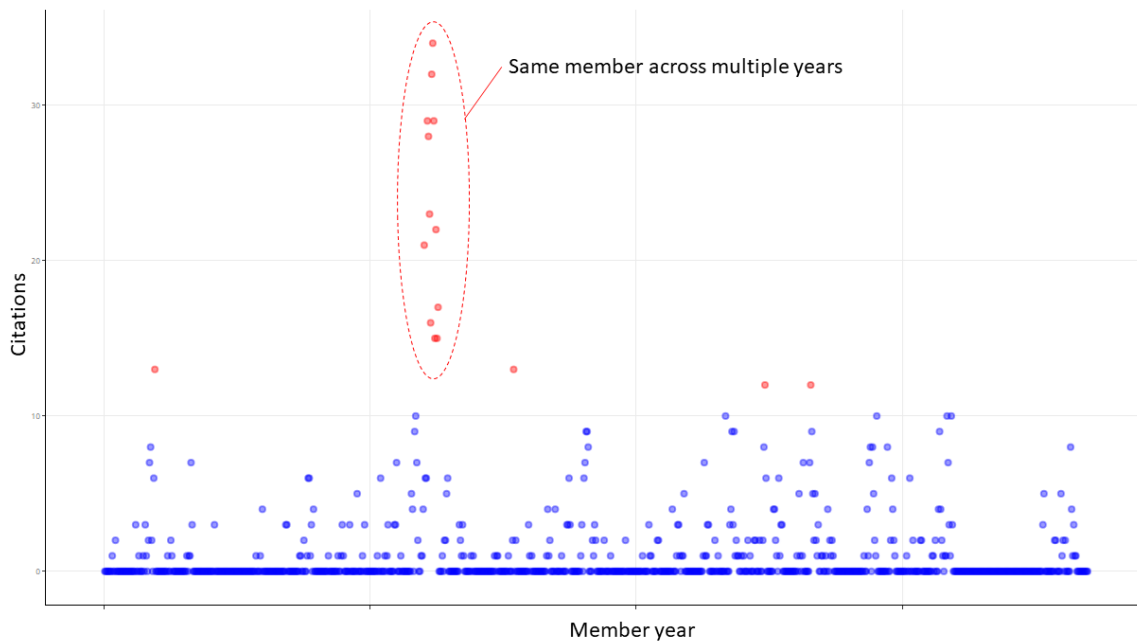


Figure 7.19: How often each SPIRE PC member (1993-2020) was cited by each SPIRE where they were PC members.

As validation we investigate if the outlier found in the analysis is correctly identified as an outlier. We can look at validation from a data acquisition and integration perspective.

Data acquisition. Validation of the front matter parsing is described in Section 7.2.5: **Validation**. For other sources, we consider the datasets as given. These datasets are considered the building blocks for this research (as they are described in Chapter 3). For the outliers found during analysis we manually verified that the data was correctly parsed from the PDF files. This was indeed the case. We also noticed that this person was also related to other conferences for SPIRE, which were not correctly loaded because of a parse error. The analysis is based on the whole population across all years. Our definition of an outliers can differ if we change the scope (e.g. '7' as threshold). However, as mentioned, this for a domain expert to decide.

Integration. We took an optimistic approach in joining the different datasets together because of the limitation in the data described in Chapter 5. It is therefore likely we miss some data. The effects of missing data could be:

- Unmatched members in the authorset because the names differs;
- Too many authors mathed to a member because of the same name (for the outlier we found we verified this was not the case);
- Unlinked referred papers because of the lack of DOI of the referred paper (the papers from LNCS do all have a DOI provided).

7.4. CASE STUDY CONCLUSION

In this chapter we looked at the PC Members and how many times their work had been referred in their proceedings. Despite the above described threads to the validity, we are able to identify outliers. From the 483 members involved in the Spire proceedings, we were able to identify 1 notable author. Depending on the threshold to be set by domain experts, this number may increase. Therefore, we can conclude that integrating data containing members of proceedings helps to direct manual effort to detect possible fraud.

8

CASE STUDY 2: JOURNAL EDITORS

In this chapter we focus on journal editors (editor-in-chief and the associate editors) of ACM (Association for Computing Machinery)¹; a publisher of journals in the area of Computer Science. Just like in the previous case study, we will start with a motivation and example of an attack, proceed with the data acquisition and integration and conclude with an analysis of the data and validation of the analysis.

8.1. MOTIVATION

This case study focuses on journal editors, which are in position with great power. As previously shown in Figure 2.2 (repeated in Figure 8.1), both roles are in a position power to make or break a publication and the career of an author.

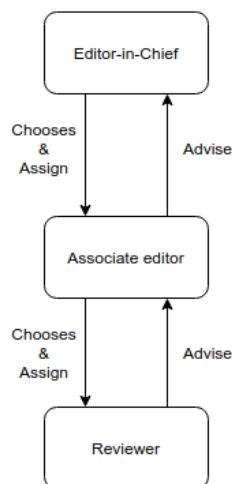


Figure 8.1: Roles within the journal process (visual interpretation of description in [Cor13])

¹<https://dl.acm.org/>

8.1.1. INTERESTING CASE: MEMBER OF THE EDITORIAL BOARD IS (CO-)AUTHOR

In this case, publications in a venue are co-authored by a member of the editorial board, which means he is publishing his own work. A questionable case in this situation is Griffiths². In Figure 8.2 we present an instance model of this situation.

Attack. We can imagine a situation in which the author is being forced to mention the person-of-interest as co-author; which is abuse of the power of the editor.

Model impact. The impact on the publication model is between the person which is editor for a venue, and the venue itself. The number of publications he has in that venue while he works for that particular venue will increase. For detection we can measure this number of publications: *number_of_publications*. Also with the case described in Section 7.1.1, we need the number of issues.

Benign alternatives. Besides possible ethical questions that can be raised, there may be a valid reason for someone to publish in the venue he is editing. For example a person is a specialist in his field and works at a venue which specializes in this specific subject. He has new work to publish but there is not another venue which captures this subject.

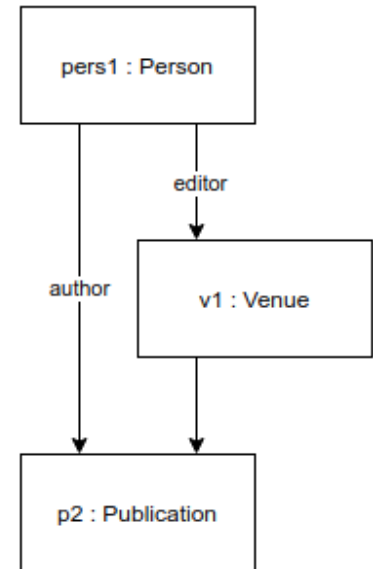


Figure 8.2: Instance model of the situation where the editor is (co-)author of the publication

8.1.2. CASE STUDY PURPOSE

As with all case studies we try to prove that adding data not yet integrated in known datasets and applying a group based outlier detection improves fraud detection. In this case, the group of focus is the editorial board. We need to compare members of this group. For comparison, we need to identify the properties that make them different and may be indicating an attack on the publication model.

As stated in the interesting case described in Section 8.1.1, a possible attack is being forced to add a member of the editorial board as co-author. One possible detection method is to see the ratio of unique authors the person-of-interest published in the journal and outside the journal. If a member publishes inside the journal with much more unique authors than outside the journal, this may indicate this member forces the original author to make him co-author. This case is based on the assumption that authors have their group of peers they publish most of the articles with. In this case study, this is the method we are going to apply.

8.2. IMPLEMENTATION

In this case study our focus is the editorial board. Editor-in-chiefs, associate-editors and sometimes a list of reviewers (generic, not who reviewed what paper) are most of the time available on the publisher website of the specific venue. But this has two issues:

²<http://deevybee.blogspot.com/2020/07/percent-by-most-prolific-author-score.html>

- *Machine readability*: These roles are not easy machine readable through an interface. It is placed on the website for humans to read. Although good technology exists to acquire data from a website, a minor change to the website can break this method and is therefore not a sustainable solution.
- *Issue- and time awareness*: Only the current editorial board is shown on the publisher website. This may be sufficient for guests of the site, but for extending the datamodel with the editorial board with the purpose of detecting fraud, this is not sufficient. In that case we need to know who was editor for which issue.

Not everything is lost however; most of the time the previous editorial boards are available in so-called mastheads or editorials of a certain issue. These mastheads can be downloaded as a PDF (as article of a journal). A formal investigation during the research preparation learned us that the format in which the editors are presented is not consistent (comparable with the front matter of the LNCS proceedings).

Theoretically, we can process the data as we did with the LNCS Front Matter. However, our intention is to prove the added value of additional publicly available data by applying an outlier detection approach within the group of editorial board members. As we already gathered data from PDF's in Case study 1, we will not repeat this technical step (we already show that this is possible) and focus more on the functional added value of having editorial board members.

We choose ACM because ACM is a big publisher in the area of computer science and they present the editorial members in a structured manner on their website for all journals. This makes acquiring the data in a automatic manner possible. To this this, we built a web scraper. In Figure 8.3 we show that, to acquire the editors (functional entity on top), we need to implement the presented flow. Details of the implementation will follow after the Figure.

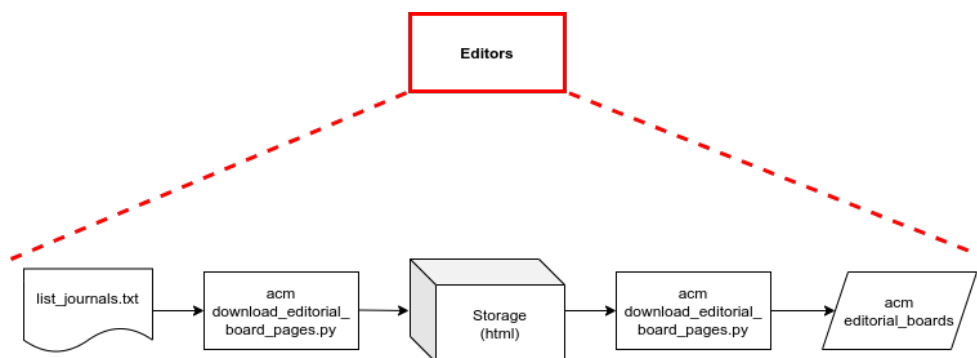



Figure 8.3: Editors acquisition

First we need to acquire the journals of ACM. Unfortunately, we were unable to find such a list on the website of ACM. Luckily, we found another website with a list of computer science journals of ACM³. We manually converted the table on this site to a textfile with urls; in Figure 8.3 show as `list_journals.txt`. Given this list, we download the pages containing the editors using the `download_editorial_board_pages.py` and store the HTML files on disk. Not working links are ignored. Figure 8.4 shows a screenshot from a part of the editorial board page of an ACM journal.

³<http://oscar-lab.org/people/~jxuan/page/resource/acm-jour.htm>

Editorial Board


Editor-in-Chief



Albert Zomaya
University of Sydney
Australia

✉
🌐


Senior Associate Editors



Hussein Abbass
University of New South Wales
Australia

Swarm Intelligence, Swarm Robotics, Explainable Artificial Intelligence, Interpretable Artificial Intelligence, Transparent Artificial Intelligence, Trust, Human-Autonomy Teaming, Human-Swarm Teaming.


✉
🌐



David Atienza
Swiss Federal Institute of Technology Lausanne (EPFL)
Switzerland

Embedded Systems, Electronic Design Automation (EDA), Edge AI and IoT Architectures, Multi-Processor SoC, Low-power Electronics, Thermal-Aware Design, Wearables


✉
🌐



Elisa Bertino
Purdue University
United States

Security, Privacy, Data Management

✉
🌐



Flavia Delicato
Fluminense Federal University
Brazil

Internet of Things, Middleware, Adaptive Systems, Edge/Fog Computing, Wireless Sensor Networks

✉

Figure 8.4: Page with editorial board from ACM (<https://dl.acm.org/journal/csur/editorial-board>)

An editorial board page consists of ‘sections’. In the case of Figure 8.4, the sections are ‘Editor-in-Chief’ and ‘Senior Associate Editors’. These sections are defined by an HTML `<h3>` tag with ‘section__title’ as class name. In case of the Senior Associate Editors, we can visually identify that the editors are placed in a grid of rows and columns. In the HTML these are represented as a row (a `<div>` with ‘row’ as class) with columns (‘col’ as class). However, the actual editor information is all located in a `<div>` with ‘profile-meta’ as class, so the ‘col’ is not needed for parsing. This construction of the page defines the logic of the parsing application:

1. Search for tags with class ‘section__title’ (use the content as role description);
2. Under this section tag, collect all rows until another section tag is found, or the end of the document is reached;
3. Within these rows, search profiles;
4. Get information from these profiles.

In Listing 8.1 the HTML of profile information is shown. All necessary information is in a `<div>` with class ‘item-meta__row’ (except for the country). Unfortunately, this page does not contain descriptive elements (e.g. ‘editor-name’ as class values). Therefore, we only can rely on tags like `<h4>` to determine the name. This is very error-prone, if the publisher decides the name should be in another tag (e.g. `<h3>`) we are unable to acquire the name.

Listing 8.1: HTML of profile information ACM (removed unnecessary parts for readability)

```
1 <div class="profile-meta">
2   <div class="item-meta__info">
3     <div class="item-meta-row">
4       <h4 class="item-meta-row">Elisa Bertino</h4>
5     </div>
6     <div class="item-meta-row">
7       <p>Purdue University</p>
8     </div>
9     <em>
10      <span style="font-style: italic; font-size: 14px;">United States</span>
11    </em>
12  </div>
13 </div>
```

The resulted attributes we acquire from the people on these pages are: role, name, affiliation, country and journal name.

8.2.1. FORMALISING THE DATASET

As stated in Section 8.1.2 we want to have the number of unique co-author a member publishes inside and outside a journal. For this we need the following entities:

- Editors
- Articles written by these editors
- Co-authors of these articles
- The venue where these articles were published

We already acquired the editors from the ACM website. The other entities can be found in the DBLP dump. However, the venue is not straightforward in the dump of DBLP; it is encapsulated in the key of an article, see Listing 8.2 as example.

Listing 8.2: Example DBLP key

```
journals/tomccap/Wang21
```

In case of the example, tomccap is the abbreviation of the journal. However, the name we get from ACM website is only the full name of a journal. In this case 'ACM Transactions on Multimedia Computing, Communications, and Applications'. More interesting; this full title is also abbreviated as tomm; thus the relation between a journal name and abbreviation is not 1-on-1.

JOURNAL ABBREVIATION LOOKUP

We need a dataset containing the relationship between the abbreviation and the full journal name. Two sets were found:

- *Pages at Clarivate*: which are the maintainers of Web of Science. Unfortunately, the abbreviations used by DBLP were not found in this set.
- *University of British Columbia*: also maintains a list. However, the abbreviations they use are different. E.g. in case of the example, their abbreviation is 'ACM Trans. Multimedia Comput. Commun. Appl.'. DBLP also abbreviates journalnames this way, but the abbreviations from this university are not found in DBLP.

The list of journals we want to scrape from ACM is limited: only 27. So a manually built dataset is a valid solution. This is very achievable with help of the search functionality on the DBLP site. See Figure 8.5 as example; the abbreviations used are shown directly right to the name.

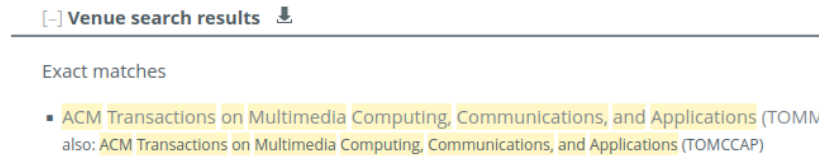


Figure 8.5: DBLP search result (<https://dblp.org/search?q=ACM+Transactions+on+Multimedia+Computing%2C+Communications%2C+and+Applications>).

INTEGRATION MODEL

With the information from the previous section, we can now build the dataset to perform our analysis upon. The dataset formulation is shown in Figure 8.6. The explanation will follow after the image.

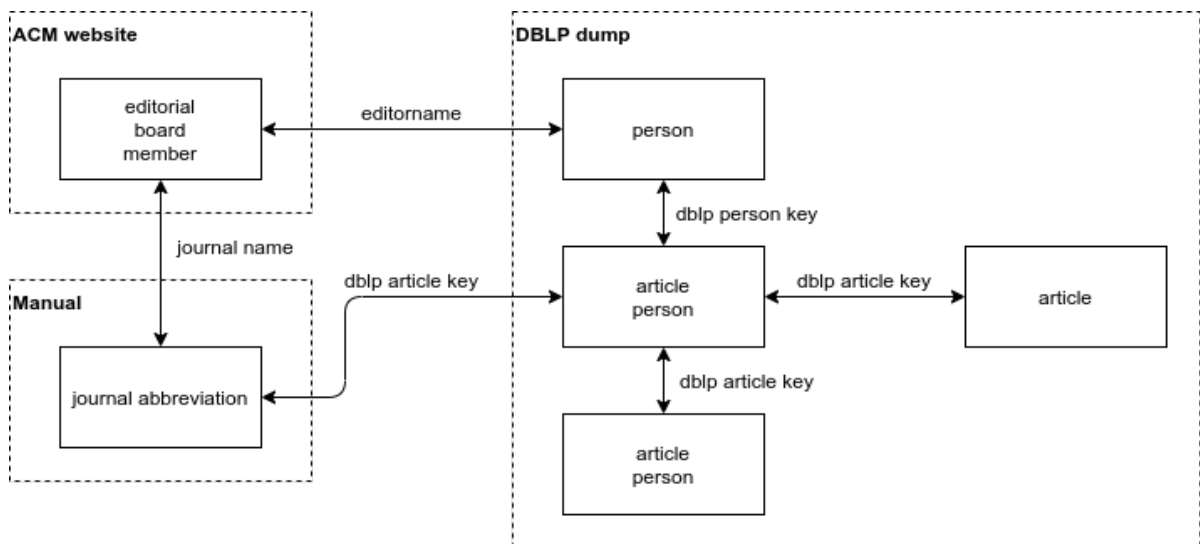


Figure 8.6: Formalising the publishing dataset

We start with the editors from the ACM website. By joining the `editorname` to the `authorname` in DBLP, we can get the unique DBLP name key (because of the issues with people covered in Section 5.3.2). With this key we can get all the articles written by the editor from the ‘article person’ table. By joining again with the ‘article person’ table on the article key we can get the DBLP person keys for all co-authors.

To know if an article is published in the same journal the person-of-interest is member of, we need to identify if the journal from the ACM website is the same of the article published. In the previous section we elaborated on the issues involved here. By using a manual dataset as link table, we can identify if the article is published in this particular journal. For this scenario we extracted the journal abbreviation from the DBLP article key. Additional information about the article can be achieved from the article table.

8.3. ANALYSIS AND VALIDATION

The properties of the editors we work with are *Number of unique coauthors inside the journal* and *Number of unique coauthors outside the journal*. From the dataset we removed publications the editor wrote himself. In the number of unique authors, the editor is excluded. In Figure 8.7 we plotted the editors. On the X-Axis the number of unique coauthors the editor has worked with inside the journal, on the Y-Axis the number of unique coauthors the editor has worked with outside the journal.

For the purpose of this case study (described in Section 8.1.2) interesting cases should show up in the right bottom (high number of unique coauthors inside the journal, and low outside the journal). To clearly show the ratio we draw a diagonal on $x = y$. In area with the red background the number of coauthors inside the journal is higher than outside the journal. The highlighted observations will be described after the figure.

Observation 1. This editor worked with 15 unique authors on 15 publications inside the journal and 15 unique authors on 18 publications outside the journal. This means with all coauthors, he publishes one article inside the journal, which is the expected behaviour. Outside the journal, we notice 2 authors he wrote more than one article with. But on a population of 15 coauthors, this is not remarkable.

Observation 2. Observation 2 has almost twice the number of unique coauthors outside the journal. Although observation is far to the right, compared with the population, this would not be a priority case to investigate.

Observation 3. We highlight this editor because the number of unique coauthors inside the journal significantly exceeds the number of unique coauthors outside the journal. This editor worked with 16 unique authors on 5 publications inside the journal and 9 unique authors on 3 publications outside the journal. In the period under investigation, there is no coauthor with whom this editor has published both inside and outside the journal. However, if we extend the period to include 2019, we find 3 scientists that have coauthored papers both within and outside the journal. Further analysis shows that this editor only publishes once with each of his colleagues, except one. Such behaviour is in line with the above described fraud – though it should not be mistaken as evidence for it. Indeed, this pattern holds for both coauthors within

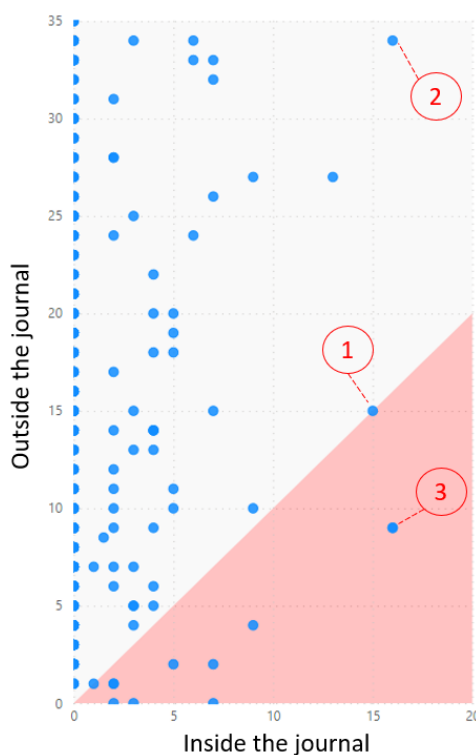


Figure 8.7: Number of unique coauthors inside vs outside the journal per editor

and outside the journal. The latter could serve as a contraindication of fraud: outside the sphere of influence of the editor, 'regular' scientific behaviour is expected. In that case, authoring only once with each colleague would be expected for this editor. Further investigation is needed to determine the cause of the observed behaviour, fraud or a more benign explanation.

8.4. CASE STUDY CONCLUSION

During this case study we clearly notice skewness in the number of unique coauthors between inside and outside the journal for a few editors; the notable observations were not exceptional at further analysis. However, if we compare these observations with other editors in the population, they do stand out. Possible explanations are:

- *No strange behaviour*: There is no strange behaviour within the group of editors.
- *Limited dataset*: We only use data of 2020 for a few journals of ACM. This set may be too limited and does not represent all editors. Outliers in the analysed dataset may be outliers within ACM, but may not be an outlier within a larger population.
- *Wrong analysis*: If authors publish multiple times and they are obliged to add the editor as coauthor, this will reflect in the analysis.

The fact they do not stand out, is not that important for the conclusion considering the goal of this research. We were able to direct the attention to a limited number of cases for further investigation; from 479 authors to 3 for the year 2020.

9

CASE STUDY 3: AUTHORS

In this chapter we dive into the authors. More specific; we look at the timeline properties of a publication.

9.1. MOTIVATION

The motivation to investigate this group from a timeline-perspective is that fraud had been detected using these properties. Also other research working with timelines are successful [SNC⁺21]. Scanff et al. investigated the relationship between the publication-lag and prolific authors, which may be a member of the editorial board. This study makes clear that dates about the publication process are interesting features. We will proceed with an interesting case involving the timeline properties.

9.1.1. INTERESTING CASE: AUTHOR REVIEWS HIS OWN WORK

An author that reviews his own work is of course a major attack on the integrity of the publication process. It should never be the case that an author becomes in a position where he can review his own work. Unfortunately, we know of such case involving Hyung-in Moon¹. In Figure 9.1 we present an instance model of this situation. We have a person (pers1) who has multiple roles in relation to one publication (p2).

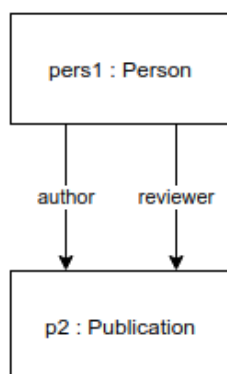


Figure 9.1: Instance model where author is also the reviewer

¹<https://retractionwatch.com/category/by-author/hyung-in-moon/>

In theory this situation can easily be detected. However, besides availability of the necessary data, the person executing such attack, will most likely not give his own name as reviewer at the moment of submitting his paper (it is normal to identify possible reviewers at submission [Cor13]). The main question here is, how are we able to identify the author and reviewer as the same person.

We approach this situation from a timeline perspective, which is how Moon is detected. A paper goes through certain stages until it is published. In Figure 9.2 we draw a timeline with the milestones of the publication process. *Submitted* is when the paper is received by the venue. *Revised* is the moment a paper is submitted again after review with changes; the asterisks implies that this can occur multiple times, or none at all if the paper is accepted at first submission. *Accepted* defines the state when the paper is ready for publishing. *Published* is the last step and is the date the paper is published. Most publishers also work with an *Available Online* date. Once accepted, a paper can be placed on the website of the publisher before the journal has been published.

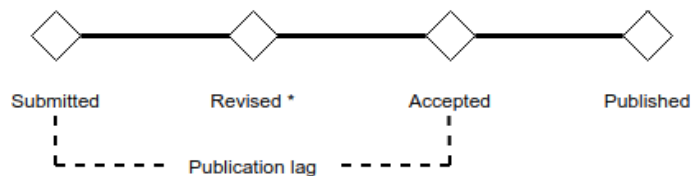


Figure 9.2: Milestones in timeline publication model with publication lag

The publication-lag is defined as the time between submission and acceptance. This measure can be used as comparison with other authors within the same venue. A low publication-lag compared with 'normal' can be used to identify interesting cases like an abnormal short time of review. E.g. Moon was detected because the time his work was submitted and approved by the reviewer (which was Moon himself) was remarkable short.

Benign alternatives. The possibility exists that a paper is that good, no revision is needed and the review did not take that much time. This also results in a low publication-lag.

9.2. IMPLEMENTATION

In this section we describe how we acquire the data necessary for this case study.

The publication date is an attribute well presented in public datasets. This date is derived from the publication of the venue. To get the dates needed to calculate the publication-lag (submitted and accepted), we need other sources. ACM and Elsevier show the dates on their website (as example for ACM, see Figure 9.3). However, these properties are not available for all publications.

■ Publication History

Published: 8 June 2021
 Accepted: 1 December 2020
 Revised: 1 November 2020
 Submitted: 1 March 2020

Figure 9.3: Publication history of an article in ACM (<https://dl.acm.org/doi/10.1145/3442445>)

As we already used ACM in the previous case study; in this case study we use Elsevier. Besides we already used ACM, we have proved to be able to extract the dates from the Elsevier website in the research proposal. When Elsevier presents an article on their website, a JSON document with details about the article is available in the HTML document. Among other details, this JSON contains a date section, see Listing 9.1 (page 61). From a functional perspective, we need the journals of Elsevier. From these journals we need the articles. This dependency is shown in Figure 9.4.

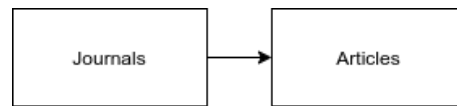


Figure 9.4: Functional overview acquisition Elsevier

Therefore, first step is acquisition of the journals of Elsevier.

9.2.1. JOURNALS

Elsevier presents a list of journals they publish on their website when using the search function. To get the journals we built a webscraper. In Figure 9.5 we position this tool in the acquisition flow.

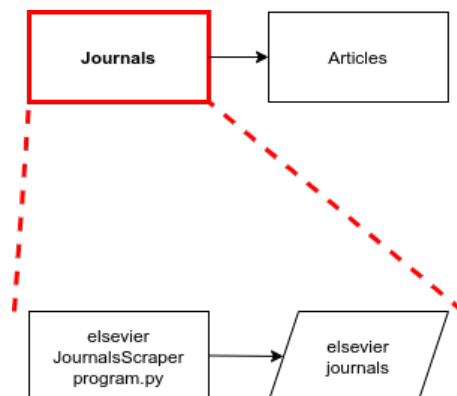


Figure 9.5: Acquisition of the journals

The search functionality of Elsevier presents the journals in a paginated list (first request returns first 100, second request 101 till 200, etc.). We can acquire the ‘next results’, by passing the page number as url parameter. By using BeautifulSoup, we can parse the resulting HTML page and load required information of the journal to the database (issn, title, href). The href is the link to detailed information (e.g. the articles published). The result is information of 2,970 journals. For our purpose we manually selected journals which we want to acquire article information here.

9.2.2. ARTICLES

From these journals we acquire the articles. In Figure 9.6 we show this in the functional landscape. We explain the details of this flow after the figure. The first step for acquiring the article information, is downloading this information. We built an application for this: `DownloadArticleMetadata`.

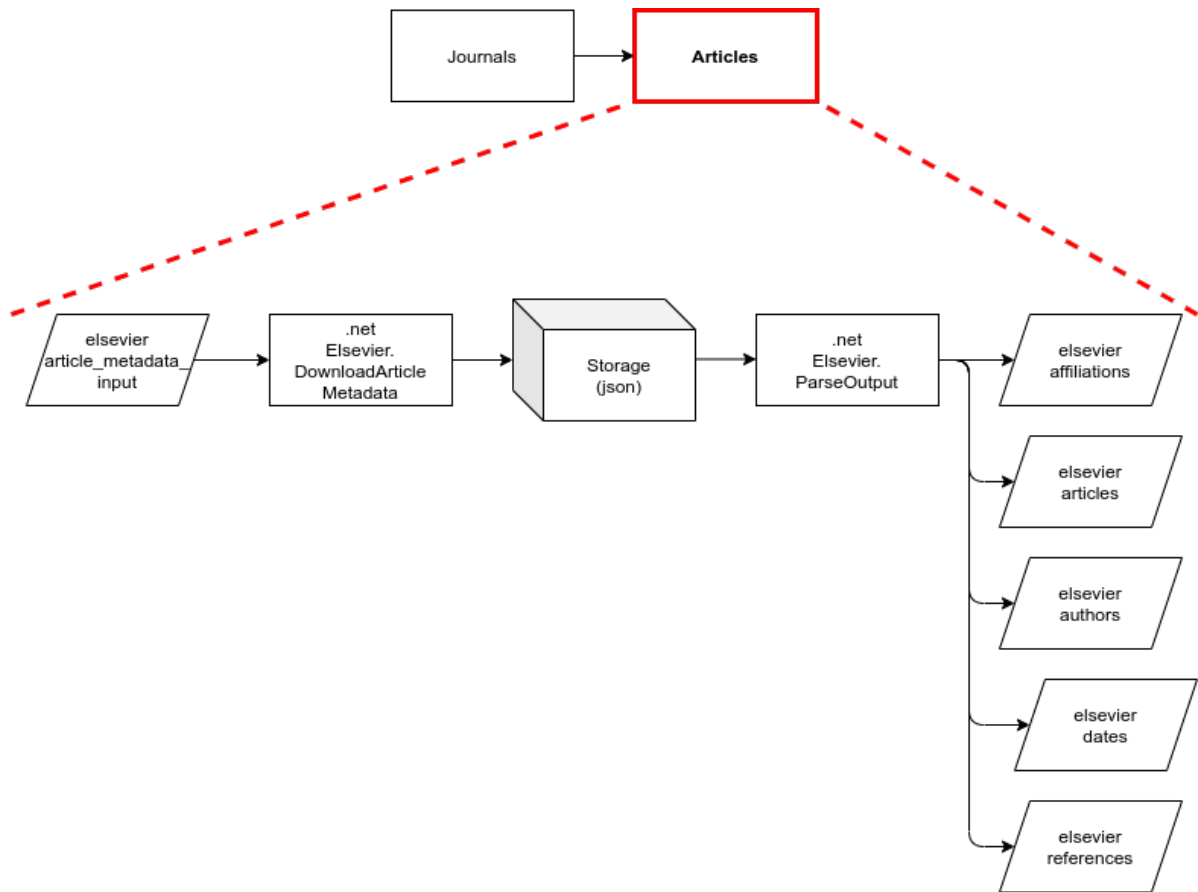


Figure 9.6: Acquisition of the articles

Again, we use the search functionality of the website. As search parameters we use the name of the journal, FLA as article type (this filters the results on research articles) and we sort by date (descending order). The search functionality only returns 1,000 results; by using the sort function, we acquire the 1,000 most recent articles.

The results of the search functionality are returned asynchronously. This means the site is presented (e.g. with a loading image) while the browser sends another request to the server to get the search results. This means that using a request library is not possible; for this part of the solution we use Selenium (see Section 6.2). The links to the articles returned in the search results are cached. After we finished iterating through the search results, we start acquiring the article content. This is not asynchronous, so this is preformed with a request library.

The HTML of an article on the website of Elsevier contains an JSON part with information about the article. In Listing 9.1 a part of this JSON document is shown which contains the dates. The JSON content is stored on disk storage for interpretation which is done by the ParseOutput application.

Listing 9.1: Dates in the JSON document

```
"dates": {
  "Available online": "25 January 2021",
  "Received": "16 July 2020",
  "Revised": [
    "14 December 2020"
  ],
  "Accepted": "4 January 2021",
  "Publication date": "25 January 2021"
}
```

The ParseOutput application iterates through the JSON files on storage, acquires information from these files and stores the information in the database. This results in five tables: articles, authors, affiliations, dates and references. As with the OpenCitations application, this tool is also multi threaded, multiple files can be processed in parallel, to improve performance.

By using web scraping and HTML interpretation we extracted the JSON documents with metadata of articles across 36 journals from Elsevier (Elsevier runs 2,970 journals). This resulted in a total of 35,136 documents. From the 35,136 we were unable to calculate the publication-lag for 4,512 (approximately 13%) because the date properties needed are missing.

9.3. ANALYSIS AND VALIDATION

In this section we present a possible method to analyse this data. In Figure 9.7 we show the distribution of the publication lag. The average is shown with a vertical red line, the median with a black line. Every dot represents a publication.

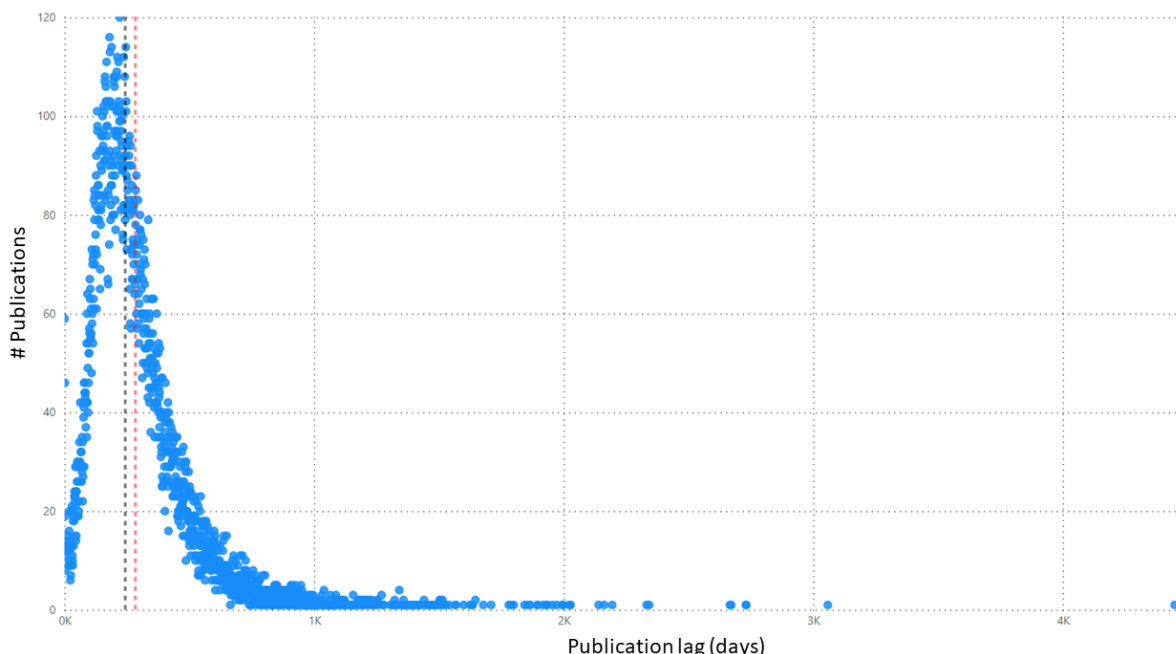


Figure 9.7: Distribution of the publication lag (median: black, average: red.)

Because of the right skewness in this dataset, we transpose it with \log_{10} to create a set

that is more normally distributed. In the publications we scraped from Elsevier, we noticed 59 observations with a publication-lag of 0 (zero). Possible explanations are:

- *Data quality*: It can be that this is an error on the side of Elsevier.
- *Actually 0 publication-lag*: Another explanation is that the publication lag is actually 0. This means the article is submitted and accepted on the same date.

We need to make a choice what to do with these observations because we can not calculate the log value of a 0 publication-lag. Considering the domain we work in it is safe to identify 0-values as outliers, because it represents a very fast process from submission to acceptance, at least within one day. To make sure these values are indicated as outliers, we replace $\log_{10}(0)$ with 0. The result of this transformation is shown in Figure 9.8. We notice that the average line and median are much closer.

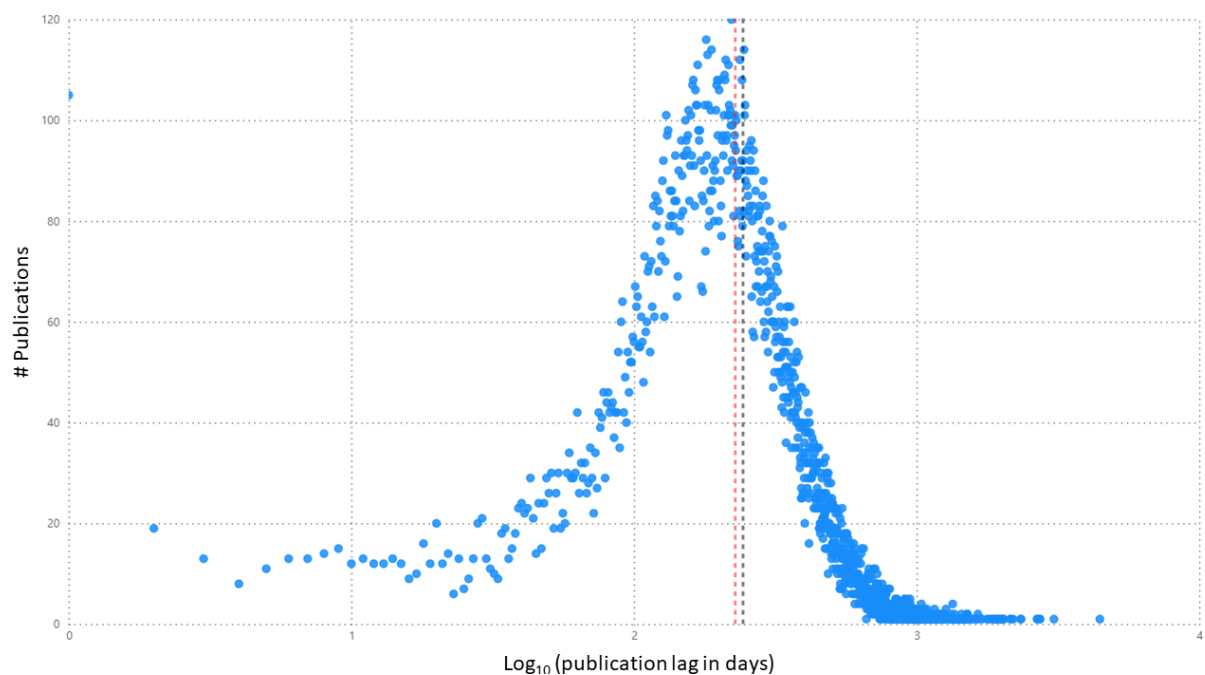


Figure 9.8: Distribution of the log of the publication lag.

In Figure 9.9 we plotted the log-values of every publication-lag, ordered by author for the ACM journal Information and Software Technology². Every dot represents a publication for one author. The log of the publication-lag on the y-axis. The x-axis is an order by name of the author (for explanation see Section 7.3). The red dots are identified as outliers (z-score < -3, 26 days or less) of all publications (not limited to this journal). The red circle in the left lower corner, we identified a 'cluster'; one author of which the publication-lag was two times very low; 8 and 11 days (average 270, standard deviation 115).

In Figure 9.10 (page 64) we compared the author of this cluster with the population and an authors with more 'normal' publication-lag duration. The publication-lags of the person-of-interest is outlined with a red dotted box (the red striped around the two observations are the same as in Figure 9.9). In the green dotted-box we plotted the publication-lag times of an author which behaves 'more' normal. An investigation of a possible relation

²<https://dl.acm.org/journal/inst>

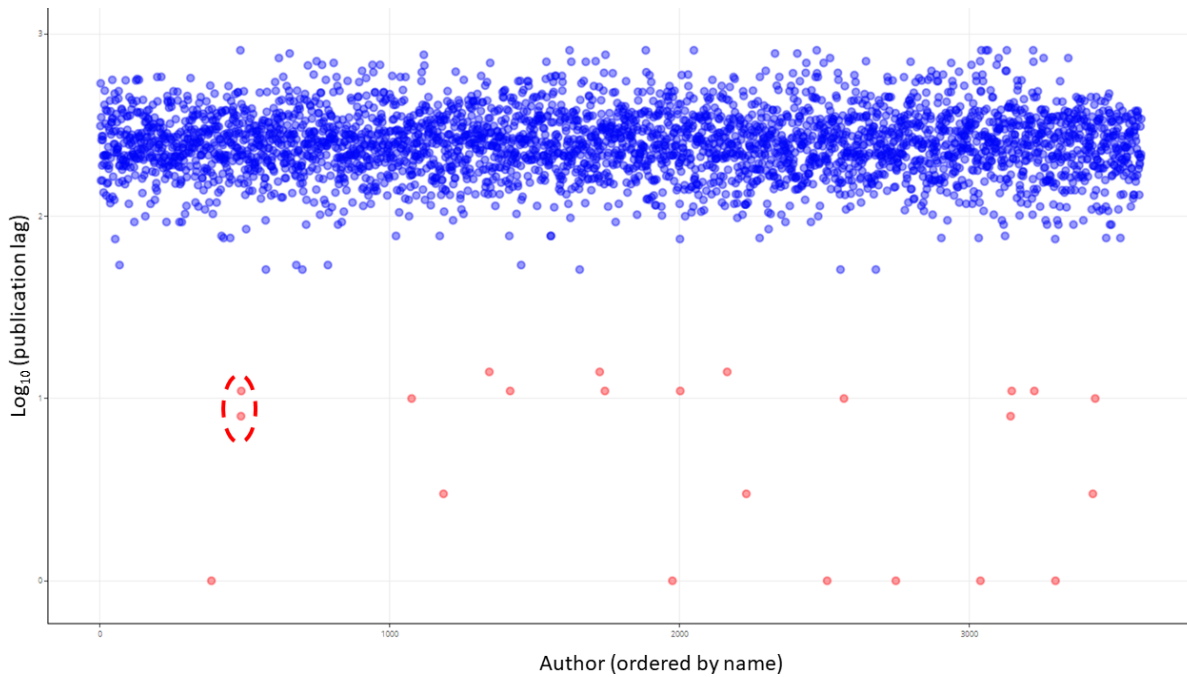


Figure 9.9: Publication lag of authors of Information and Software Technology.

between the person-of-interest and any member of the editorial board did not return notable results. The authors of these papers were not part of the editorial board, nor did we notice an editors of the same affiliation of one of the authors.

9.4. CASE STUDY CONCLUSION

In this chapter we took data of Elsevier for a a subset of journals in the domain of Computer Science. Almost 34% of the data we acquired from Elsevier was not usable for analysis. We investigated one journal and were able to identify a person-of-interest with two publications of which the publication-lag deviates from other publications in the journal.

At this point we can conclude that we notice some very low, even zero, days between submission and acceptance of a publication. We are not able to say anything about the outliers. Further investigation can combine these properties of the publication with data of editors (e.g. is the author an editor, is the editor a colleague of the author). Another direction to investigate can be a possible pattern in reviewers and publication-lag. We did not found a source to get data of reviewers for a certain publication, but most of the times the reviewers are available at venue level (e.g. in the masthead).

Considering the goal of this research; of all scraped articles of Elsevier of which a publication-lag could be calculated, we were able to direct the attention from 30,624 articles (written by 70,822 authors) to 273 articles (written by 878 authors), if we take a threshold of 14 days. That is 0.89% of the articles and 1.24% of the authors.

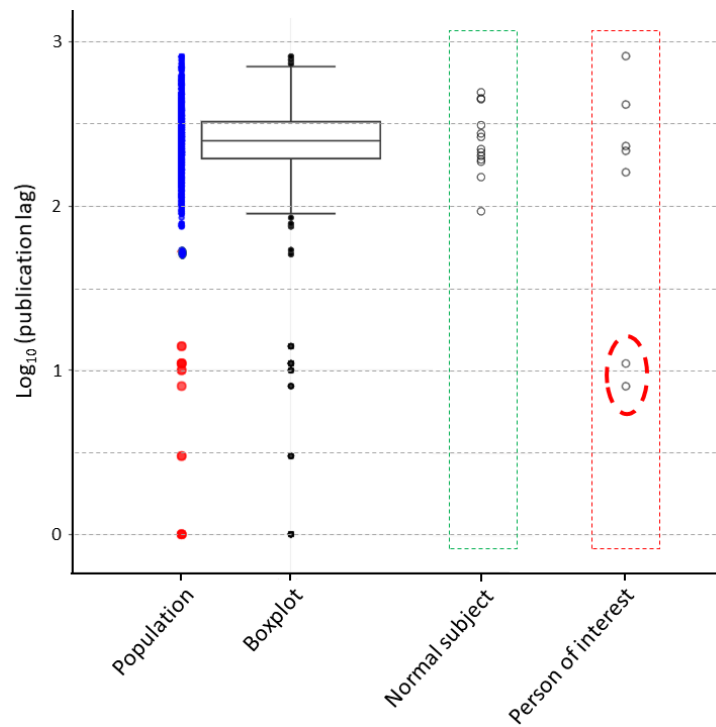


Figure 9.10: Log of the publication lag of population compared with person of interest and more 'normal' author.

10

CONCLUSIONS

In this research we set out to direct manual investigation to notable cases. We conclude this research with the following key findings:

Lack of data quality limits creating an integrated dataset. Source depended keys (e.g. for identifying authors) are only applicable within that single dataset. Therefore integrating datasets should use source-independent key attributes (DOI and ORCID). Ideally datasets containing data of the publication process should include these attributes. In practice the quality and availability of these source independent keys is not sufficient for identifying objects (see Section 5.3). This impacts the quality of the resulting integrated dataset.

Alternative approach to use names instead of ORCID for identifying objects is not sufficient. However, it is possible to use alternative approaches (see Aminer [TZY⁺08]); achieving a high enough level of quality requires a lot of effort. For the purpose of our research, we therefore conclude that creating an integrated dataset for analysis purposes is too cumbersome.

Enriching standard datasets is made more difficult because of diffusion coupled with lack of structure of data. Standard available datasets are based around the author, paper, venue and sometimes the citation. These datasets are insufficient for generic outlier detection [Tie17]. Standard datasets miss important data (e.g. data of PC members, publication-lag). In theory this data is publicly available, mostly on the website of publishers. In practice, this data is in an unstructured form. Due this unstructured form, acquisition of this data and integration with existing datasets is challenging. For example, our proof-of-concept PDF data extractor, specifically tailored to its source material, still only achieved a success rate of ~47% for extracting data.

Applying group based approach on enriched data yields useable results. We executed three case studies to investigate if applying a group based approach can indicate notable cases for manual investigation.

In the first case study we gathered the PC members from LNCS front matter documents and combined this data with DBLP, OpenCitations and scraped data from Springer Website. As an example analyses we looked at how often these members where cited in their own journal. For one proceeding we plotted the results in Figure 7.19 (page 47). Depending

on the threshold, we limited the scope for analysis from 483 member of the SPIRE proceedings to 1 outlier.

In the second case study we acquired the Editorial board of ACM journals. This data is combined with DBLP to get the co-authors. As a possible analysis method, we looked at the number of unique co-authors these editors worked with inside versus outside the journal. In Figure 8.7 (page 55) the results of one journal are shown. Although not as clear as case study 1, we are able to direct manual investigation from 479 authors in total to 3 notable cases (especially observation 3 in Figure 8.7).

As third case study, we focused on the time aspect of the publication process by looking at the publication-lag. In this case we did not integrate additional data. Instead of indicating the value of additional objects, we show the value of additional attributes for already available objects in standard datasets. We used data of some journals we scraped from Elsevier. In Figure 9.8 (page 62) we see the deviation of the publication-lag. On the left side we notice very low publication-lags. By applying a threshold of 14 days, this approach limits the subjects for investigation from 30,624 articles (written by 70,822 authors) to 273 articles (written by 878 authors).

The question we aim to answer during this research was: *To what extend can integration of publicly available data sources contribute to directing manual investigation of scientific fraud?* As answer we can formulate is that integrating publicly available data enables a group based outlier detection approach, which yields notable cases for further manual investigation.

Impact of this research. The conclusions and part of the approach of this study can result in a detection framework that directs investigation of fraud to the most striking cases. This improves the detection of fraud cases which eventually results in a more fair scientific publication environment.

10.1. FUTURE WORK

We present three directions for further research: Data acquisition, Data integration and an alternative detection method.

10.1.1. DATA ACQUISITION

This research shows the added value of additional acquired data; PC members from Springer LNCS, Editors of ACM and publication-lag of Elsevier. However, these datasets has their weak spots which could be improved. In this section we propose some improvements in data and process.

Editorial board. The added value of having data about the Editorial board has been mentioned multiple times by other researchers. In our research we used webscraping to get the editorial board of a limited number of journals from ACM. Our approach has a few weak spots:

- The data is limited to journals of ACM;
- Only the active board is acquired.

A solution that would acquire editorial board documents (comparable with the LNCS front matter documents) and parses these documents, would deliver high added value to the research in scientific fraud.

Review data. Data about the review process is not publicly available. However, if we look at the publication process, this review part has a high impact on the scientific performance of authors. Therefore, data about this process is valuable. Open initiatives to review papers do exist. One of them is Pubpeer¹, where researchers can comment on papers. Mining this data adds value about the quality of publication process of venues.

Industrialise data acquisition. In this research we acquired the data as a one-time effort. This results in pieces of a possible end-to-end data pipeline (from acquisition to analysis). When directing for a general safety net, a more structured approach should be set up. We propose two directions necessary for this safety net:

1. The pieces of the pipeline should be chained together to form a a more streamlined process;
2. Loading complete datasets every time is cumbersome. Therefore acquiring deltas could improve performance of the detection process.

Front matter parsing. From the validation of the front matter parsing proof-of-concept (Section 7.2.5, **Validation**) it is clear that improvements are possible. This will result in a better parsing and therefore in more and better data. However, this is still a quick win: only name and affiliation are available in these documents and formats can and will change. Our assumption is that continuing this approach, the amount of acquired data will not exceed the 80% of available data in the front matter documents. Therefore, the need for a machine readable standard (see recommendations) stays, but this direction does improve the analysis on short-term; resulting data will certainly be better than what is available right now.

Also, besides improvement of the data acquisition tool, research could be conducted if the same approach and software can be applied to load documents from other publishers (e.g. IEEE).

10.1.2. DATASET INTEGRATION

In this research integrating multiple datasources to create one dataset was not feasible. One reason is that people involved in the publication process were not uniquely identifiable. Possible directions to improve the integration of people are:

ORCID as source. In this research we approached integration directed from the data-sources, which may contain an ORCID for the author. In Figure 10.1 this is drawn on the left side. Another approach may be to see the ORCID as source, and match people based on their publication (in ORCID people can add their publications) *or* ORCID, shown on the right. In this new situation, ORCID becomes a bridge between the sources.

¹<https://pubpeer.com/>

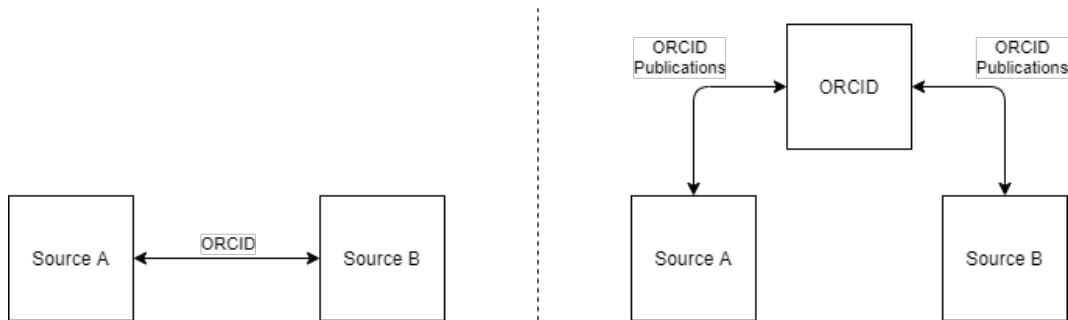


Figure 10.1: Left: our approach, right: proposed improvement

10.1.3. ALTERNATIVE DETECTION METHOD

The focus of this research is on the added value of integrating additional data sources. The case studies to prove this value were all executed with attacks in mind. Therefore, the datasets created for this analysis were focused on certain metrics to indicate these attacks. The weakness is that this approach is still specific to each attack: for every new type of attack discovered, a new analysis or detection method has to be implemented. To discover these new attacks, the community ends up relying on whistleblowers, which essentially brings us back to square one; no generic approach. Our proposal is to look at the structure of the data from a graph representation. In Appendix B we will perform an initial exploration on this approach.

10.2. RECOMMENDATIONS

The most recommendations we present are for the publisher.

Machine readable standard. As we shown in this research, currently valuable data is available, but:

- In an unstructured manner (some in PDF documents, e.g. editors, some data on website e.g. dates);
- Every publisher provides this data differently.

This results in problems acquiring this data:

- Lots of publisher-fitted procedures should be implemented;
- It is not possible to separate between data not known, or data not presented.

This last point is important. Consider the following two structures to represent a person. In Listing 10.1 (page 69) the ORCID element is provided, but it is not known. The receiver (client requesting this data) knows that the ORCID is not known by the sender (which is eventually also information about data-completeness). In Listing 10.2, it is unknown if the sender knows the ORCID, but is not presenting it, or if the sender don't know the ORCID, and therefore not sending this value. Therefore, our recommendation is to define a machine-readable standard for data exchange with no optional fields (values can be null). With a standard in place, more research based on improved data can be conducted, which eventually leads to more fraud detected.

```

"Person": {
  "FirstName": "Jodocus",
  "LastName": "Kwak",
  "ORCID": null
}

```

Listing 10.1: Example of not known data

```

"Person": {
  "FirstName": "Jodocus",
  "LastName": "Kwak"
}

```

Listing 10.2: Example of not presented data

Apply fraud detection on the process. Currently automatic fraud detection procedures are in place for publication content, e.g. plagiarism. On the publication process side, most publishers have a code of conduct. However, we are unaware if automatic detection of fraudulent behavior is in place. Therefore, our recommendation is to:

- Make detection of fraudulent behavior an integral part of the publication process;
- If procedures are in place, provide information to the public that the publication process is being monitored for fraudulent behavior.

The result is that people with malicious intent become restrained in attacking the publication process.

Make more data about the review process public. Data about the review process is not publicly available. However, this review process is essential for the scientific community; this is the ‘self-controlling’ element that leads to a certain research quality. This is also a weak spot; the result of the review entirely depends on the reviewer, which may, or may not, be personally involved. Therefore our recommendation is to provide metadata about the review process, e.g.:

- Who reviewed which paper;
- When was the paper presented to the reviewer;
- When was the result of the review returned;
- What was the result;
- What methodology was applied (e.g. single-blind, double-blind).

Providing this data to the community will result in monitoring the review process on personal involvement and therefore forces the review to become more about the research instead of the researcher.

ORCID. During analysis of the scraped data from Springer, we ran into cases where an author has multiple ORCID’s. Checking these ORCID’s at <https://orcid.org/> made clear that these are indeed the same person, but working at different universities. Our recommendation for researchers therefore is to use one ORCID consistently.

10.3. DISCUSSION

In this research we found that applying group based outlier detection on enriched datasets with publicly available data can direct manual investigation to notable cases. This means that valuable data is ‘hidden’ in PDF documents and shown on websites.

The fact that we can detect outliers does not mean that we can detect fraud, we can only address outliers. As stated in the introduction, the underlying assumption of this research is that fraudulent behavior will be reflected as outliers, but not all outliers are fraud. This raises the question if we will ever be able to set up a system that detects fraud in the scientific publication process. In our opinion, we should always apply for fair hearing.

Our main goal was to propose an approach to direct manual investigation to notable cases. Theoretically, this approach can be applied for that goal. However, we are unable to determine if this actually results in finding frauds. A more comprehensive study with current 'catches' compared with investigation of outliers found by our proposed approach should provide the conclusion if this approach pays off.

We can not state that we find an enormous number of interesting cases; this depends on the threshold that defines the outlier. However, the threshold we put in place during the case studies were high. Investigating the list of outliers from most to least, will eventually reach a point that investigating does not pay off anymore. The border before that moment can be set as a threshold. Only then we know how successful our approach is in actually finding frauds.

Addressing cross-publisher fraud. Current fraud detection mechanisms typically focus on the content of the publication. However, as argued in this thesis, data on the publication process may be leveraged to uncover forms of fraud that have escaped notice so far. One example of such a fraud is 'lightweight' fraud that is repeated across many publishers. This fraud becomes impactful due to the quantity of affected works, not the impact of a single fraudulently published work. Addressing such fraud requires at the very least integration of data from different publishers.

A solution could be an independent institute which receives the necessary data to do cross-publisher analysis. This institute could serve as a provider for analytical services for publishers unable to set up this necessity themselves. The responsibility of this institute is to address and investigate notable cases. Funds for such an institute may come from institutes that gain from a fair publication-process. In the Netherlands this may be the NWO (Dutch Organisation for Scientific Research), or a partnership of universities.

BIBLIOGRAPHY

- [BH04] Bo-Christer Björk and Turid Hedlund. A formalised model of the scientific publication process. *Online Inf. Rev.*, 28(1):8–21, 2004. 5, 7, 12
- [BK17] Hannah Bast and Claudius Korzen. A benchmark and evaluation for text extraction from PDF. In *2017 ACM/IEEE Joint Conference on Digital Libraries, JCDL 2017, Toronto, ON, Canada, June 19-23, 2017*, pages 99–108. IEEE Computer Society, 2017. 12, 29
- [BL20] Mario Biagioli and Alexandra Lippman. *Gaming the metrics: misconduct and manipulation in academic research*. Mit Press, 2020. 8
- [Cor13] Graham Cormode. What does an associate editor actually do? *SIGMOD Rec.*, 42(2):52–58, July 2013. 6, 12, 16, 49, 58
- [CPN21] Joyita Chakraborty, Dinesh K. Pradhan, and Subrata Nandi. On the identification and analysis of citation pattern irregularities among journals. *Expert Systems*, 38(4):e12561, 2021. 11
- [Hau15] Charlotte J. Haug. Peer-review fraud — hacking the scientific publication process. *New England Journal of Medicine*, 373(25):2393–2395, 2015. PMID: 26488392. 10
- [HPS19] Ivan Heibi, Silvio Peroni, and David M. Shotton. Software review: Coci, the opencitations index of crossref open doi-to-doi citations. *Scientometrics*, 121(2):1213–1228, 2019. 11
- [HVWvA19] Chris HJ Hartgerink, Jan G Voelkel, Jelte Wicherts, and Marcel ALM van Assen. Detection of data fabrication using statistical tools. 2019. doi:10.31234/osf.io/jkws4. 3
- [JM17] Hugo Jonker and Sjouke Mauw. A security perspective on publication metrics. In Frank Stajano, Jonathan Anderson, Bruce Christianson, and Vashek Matyás, editors, *Security Protocols XXV - 25th International Workshop, Cambridge, UK, March 20-22, 2017, Revised Selected Papers*, volume 10476 of *Lecture Notes in Computer Science*, pages 186–200. Springer, 2017. 10, 12, 13, 16, 19
- [Ley09] Michael Ley. DBLP - some lessons learned. *Proc. VLDB Endow.*, 2(2):1493–1500, 2009. 11
- [LO15] Daniel Linstedt and Michael Olschimke. *Building a scalable data warehouse with data vault 2.0*. Morgan Kaufmann, 2015. 12, 17

- [LWN⁺20] Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. S2ORC: The semantic scholar open research corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Online, July 2020. Association for Computational Linguistics. [12](#)
- [MHWT10] Arthur M. Michalek, Alan D. Hutson, Camille P. Wicher, and Donald L. Trump. The costs and underappreciated consequences of research misconduct: A case study. *PLOS Medicine*, 7(8):1–3, 08 2010. [3](#)
- [NX14] Yong-jie NIU and Su-qin XUE. Realization of extraction of academic papers information based on pdfbox. *Computer Technology and Development*, page 12, 2014. [12](#)
- [RRB⁺10] Lars Rönnbäck, Olle Regardt, Maria Bergholtz, Paul Johannesson, and Petia Wohed. Anchor modeling - agile information modeling in evolving data environments. *Data Knowl. Eng.*, 69(12):1229–1253, 2010. [12](#), [17](#)
- [SCMM19] Marco Seeber, Mattia Cattaneo, Michele Meoli, and Paolo Malighetti. Self-citations as strategic response to the use of metrics for career decisions. *Research Policy*, 48(2):478–491, 2019. Academic Misconduct, Misrepresentation, and Gaming. [2](#), [3](#)
- [SNC⁺21] Alexandre Scanff, Florian Naudet, Ioana Cristea, David Moher, Dorothy V M Bishop, and Clara Locher. ‘nepotistic journals’: a survey of biomedical journals. *bioRxiv*, 2021. doi:10.1101/2021.02.03.429520. [10](#), [57](#)
- [Str97] Marilyn Strathern. ‘improving ratings’: audit in the british university system. *European Review*, 5(3):305–321, 1997. [2](#), [8](#)
- [Tie17] Niels Tielenburg. Automating outlier detection in academic publishing. Master’s thesis, Open University, 2017. [10](#), [13](#), [25](#), [65](#)
- [TZY⁺08] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: Extraction and mining of academic social networks. In *KDD’08*, pages 990–998, 2008. [11](#), [20](#), [65](#)
- [UYK18] Erdiñç Uzun, Tarık Yerlikaya, and Oğuz Kırat. Comparison of python libraries used for web data extraction. *Journal of the Technical University - Sofia Ploudiv branch, Bulgaria*, 24:87–92, 2018. [12](#)
- [Zha17] Bo Zhao. Web scraping. *Encyclopedia of big data*, pages 1–3, 2017. [11](#), [12](#)

Appendix A

Dataset PC Member citations

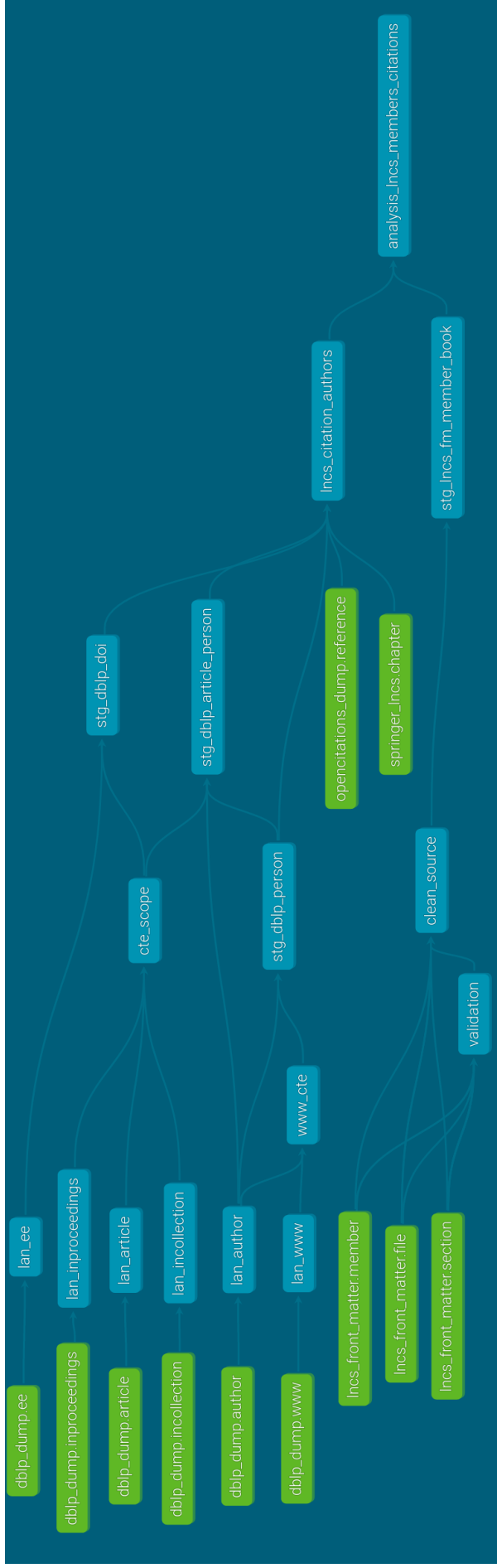


Figure A.1: Dataset composition PC Member citations

- **Green nodes:** Source tables, tables loaded by acquisition tooling.
- **Blue nodes:** Tables or views.
- **Lan prefix:** Tables with attributes in correct format and indexed for performance.
- **Stg prefix:** Tables ready for integration. More 'business like' objects (e.g. Person, Article).

Appendix B

Research Proposal: Exploration of abstract graph-based approach to outlier identification

This proposal is to find a common denominator across attacks by approaching the publication process domain from a graph perspective.

Figure B.1 shows a graph representation of self-citation. Note that this is a 3-cycle directed graph: a person (node) authors (edge) a paper (node) which cites (edge) a paper (node) written by (edge) the same author (the first node).

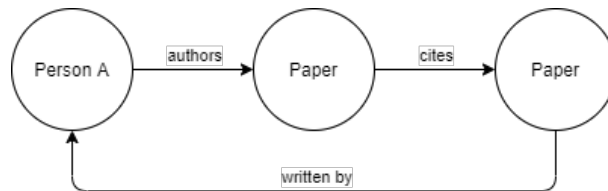


Figure B.1: Graph representation of self-citation

For readability we flatten this graph in Figure B.2. These are exactly the same situations. Person A on the left is the same as Person A on the right (dotted line); in this chapter this notation is used to represent the same node.

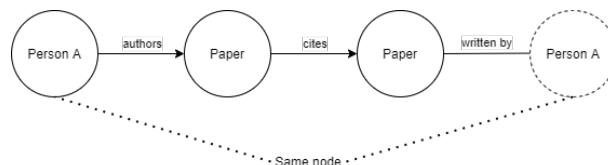


Figure B.2: Graph representation of self-citation, flatten

If a person ‘attacks’ the publication process by an exorbitant number of self-citations, this graph would extend as in Figure B.3.

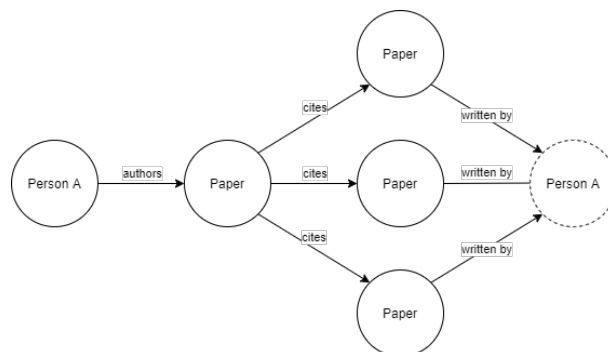


Figure B.3: Graph representation of multiple self-citations.

As such, the representation of a person that publishes in the venue he is editor of, is shown in Figure B.4. This situation can be extended as in we did in Figure B.3.



Figure B.4: Graph representation of publishing in own journal.

In this research, an outlier is defined a an observation which matches or exceeds a thresholds; we define what is considered normal behaviour and indicate outliers. Now lets consider the following situation:

Let $A1$ = self-citation attack (Figure B.2).

Let $A2$ = self-publishing attack (Figure B.3).

Let x = outlier threshold set to 3.

Let $count(A1) = 2$.

Let $count(A2) = 2$.

If we apply detection mechanisms as described in this research, both attacks are not considered an outlier (because $2 < 3$). However, for the person applying these attacks, the total of ‘gains’ he has is 4 ($count(A1) + count(A2)$). Attackers who take these thresholds in account (and therefore stay under the radar), will not be detected. This situation is represented in Figure B.5.

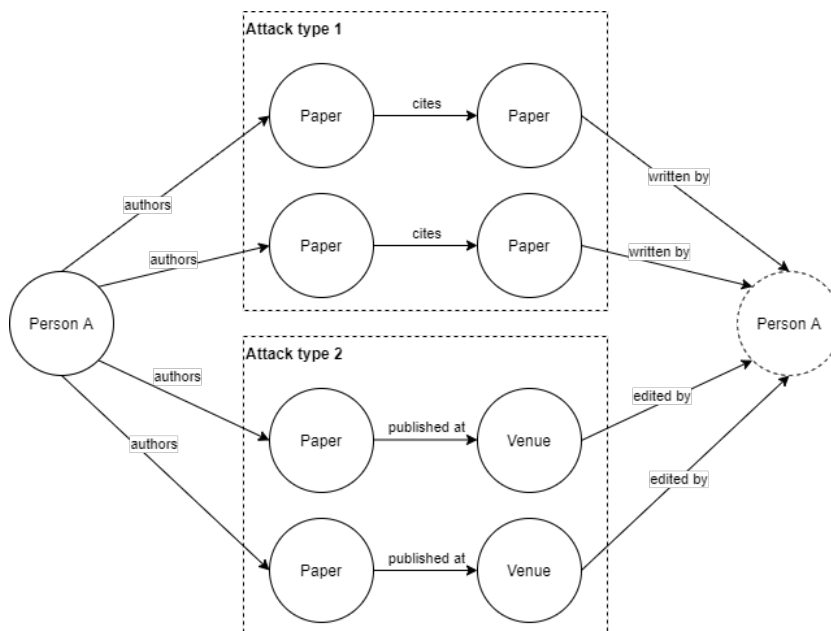


Figure B.5: Two types of attacks.

We see 4 paths, 2 for every attack; so considering the threshold x , this should be an outlier.

A person will attack the publication process to improve its own metrics. Therefore, the assumption that this future work proposal is based upon, is that an attack can be considered a path from and to the same person. An abstraction of this fraud detection approach is shown in Figure B.6; an attacker as Node A which performs N attacks, visualized by the paths 1 till N .

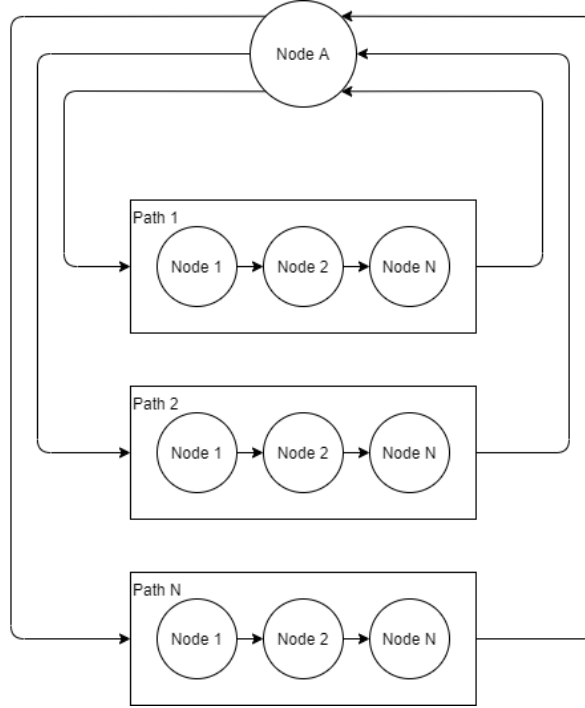


Figure B.6: Abstraction of paths.

Considering this abstraction. We can set the following:

Let $n = \text{Node}$.

Let $p = \text{Path}$.

Let $isperson(n) = n$ is of type person.

Let $outgoing(n, p) = \text{path } p \text{ is outgoing of node } n$.

Let $incoming(n, p) = \text{path } p \text{ is incoming in to node } n$.

The count of collection of paths of a person which eventually returns to the same person can than be expressed with Predicate B.1.

$$\forall n(isperson(n) \implies \forall p(outgoing(n, p) \wedge incoming(n, p))) \quad (\text{B.1})$$

Benefits. This approach has the following benefits:

- At forehand specified derived metrics are not necessary (e.g. *count of citations* or *count of publications*), the only interesting metric becomes the *count of paths*.
- *How* the publication process is being played becomes less relevant for detection. This approach indicates *that* the process is being played. Although this should not neglect the analysis of the attack to gain knowledge of attacks.

- Changing the *isperson()* to *isjournal()* or even *isaffiliation()*, abstracts the type of the attacker. With this approach attacks of journals to improve their Journal Impact Score or affiliations that behaves like a high performing research facilities can be investigated.

Caveats. This approach comes with some caveats: That fact we reason about a directed graph is important. Also, the roles of the edges should not taken lightly. For example: If a person *authors* an article, the article is *written by* that person (Figure B.7). Theoretically, this results in a cycle which will be counted in Predicate B.1.

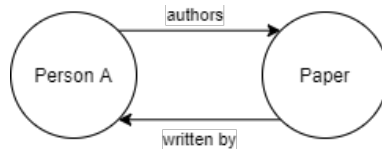


Figure B.7: 2-Path cycle to neglect.

To prevent this, some measures should be taken. E.g.:

- Set a minimum count of edges (e.g. neglect paths with less than 3 edges);
- Set a guard on the label of edges that inputs and outputs an attack.