ulm university universität
uulm

Ulm University | 89069 Ulm | Germany

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

Academic-CV

Bachelor Thesis at Ulm University

Presented by:
Stefan Wintergerst
stefan.wintergerst@uni-ulm.de

Examiner:
Prof. Dr. Timo Ropinski

Advisor:
Christian van Onzenoodt

2020

Last updated September 23, 2020

Typesetting: PDF-LaTeX 2$_\varepsilon$

Abstract

Metrics such as the h-index are used to assess the quality of an academic author or group. Those metrics mainly focus on the amount of citations the author has received and are rarely influenced by anything else. Most of the time, the citation count and the corresponding metrics are not enough to get an accurate overview of an author's career. Factors such as yearly trends, author position distribution and the community size are important to achieve exactly that, but are not easily accessible. For example, the community size (total number of unique authors of an author's journals and conferences) directly impacts his h-index due to bigger journals having an higher impact-factor. Several ways to adjust these metrics were proposed, but none of these variants are easily accessible and none of them take the authors community size or scientific age into account [2][5][6]. AcademicCV seeks to provide a tool to compare two scientific authors based off established metrics, yearly trends, author position and citation distribution and the community size with a focus on visualizing those metrics in appropriate ways.

Contents

1 Introduction

The first chapter will give a short introduction about the subject of this paper. First, we will discuss the problem and our approach to a solution (Section 1.1) and then we will discuss the structure of this paper (Section 1.2).

When comparing two scientific authors, there are several common metrics to assess an author's scientific output. Most of these metrics are numeric values such as the h-index or i10-index and almost none of them consider the author's place in the author listing or the community size. There are several variants of these metrics, such as the g-index, which averages the citations and therefore does not ignore very highly cited publications as the h-index or the work of Post et al.[5], which tries to include the authors position in the author listing within h-index. The position of the author usually indicates how much work he has contributed towards the publications with being listed first indicating that the author has done most of the work and being listed last indicating that the author has done minimal work. The community size is defined as the total number of unique authors from each journal or conference the author has published within.

This is an important metric to consider, since publishing within bigger communities inherently gives the author an advantage over authors who publish within smaller communities due to the higher impact-factor of bigger journals and the overall greater exposure.

Another problem is that, as of the time of writing, there is no tool to easily compare two authors. The comparison would require the user to manually search for an author's metrics and compare them to others. Our goal was to provide a tool that allows the user to do exactly that, to easily compare two scientific authors and have all metrics in one place. Additionally, we wanted to visualize metrics which are hard to interpret from just numbers or lists via appropriate types of graphs. We grouped those metrics into broader categories such as yearly trends, which visualize the author's number of publications and his citation count over the span of his career and indices, which display established metrics and our new one in appropriate ways. Another goal was to ease the problem of data acquisition. Most databases have problematic licensing and limitations. This is why we created Academic-CV, a website where anyone can compare two authors and where all common metrics are present and visualized. The website allows the user to compare authors by simply entering their names and the website then displays all common (and

new) metrics with appropriate forms of visualizations such as bar charts or tree maps. By using the backend of sonne[7], we also solved the issue of of data acquisition, since the dataset used within the server's database is freely licensed and the server does not have any access limitations. The remainder of this paper is structured as follows: we will further define the community size of an author and describe how we visualized other metrics in meaningful ways (Chapter 2), then we will dive into related work and metrics we included within our application (Chapter 3). Next we have to discuss how we acquired the data we use to compute those metrics and how we decided on which data source to use (Chapter 4). In Chapter 5 and 6 we discuss how the application is structured and implemented along with how and why of the visualizations we used. Chapter 7 shows a small case-study of two authors and finally, Chapter 8 provides and conclusion and possible features which could be implemented in future works.

2 Motivation

We had two main tasks we wanted to accomplish. Number one was to calculate an authors community size since none of the established metrics take it into account. Two was to create a platform that allows for easy comparison of two scientific authors based on established metrics such as the h-index and i10-index, yearly trends such as publications per year and other metrics like citation distribution or number of publications published in certain journals and conferences. Additionally, we wanted to provide visualizations for every metric, since, compared to plain numbers, a graph is generally more easily readable. The use case for this application is primarily focused on recruitment processes within scientific organizations such as post-doctorate or professor applications.

2.1 Community Size

Since different fields of study within a scientific branch (e.g. Visual Computing and Human-Computer-Interaction for media informatics) can have different community sizes, the impact on the common metrics to assess someone's academic accomplishments can be quite substantial.

That is why we propose a new metric for our first task: The community size cs of an author. Assuming an author has n journals and m conferences, cs is defined as in Equation 2.1:

$$\begin{aligned} cs_j &:= \sum_{k=1}^{n} j_k \\ cs_c &:= \sum_{k=1}^{m} c_k \\ cs_t &:= cs_j + cs_c \end{aligned} \tag{2.1}$$

Figure 2.1: The definition of the community size with n being the number of the unique journals and m being the number of unique conferences the author has published in. j_k and c_k are the number of unique authors from the k-th unique journal/conference the author has published in.

The problem of this approach is that journals where the author only has a single publication are weighted the same as journals with a higher number of publications.

Journal	Publications	CS
PUG	1	793
LNCS	**1**	**80973**
PMC	4	2918
arXiv: CVPR	4	22843
CGF	5	5793
ACM TG	5	2492
TVCG	14	7608
cs_j	34	123 420

Table 2.1: Overview of an imaginary author's community size. Journal names are abbreviated. The outlier with only one publication is marked in bold. The journal with the most publications only has 10% of the community size compared to the outlier.

As seen in Table 2.1, the journal 'LNCS' contributes more than half of the unique authors to cs_j even though the author only published one article in the journal. Additionally, almost 75% of cs_j comes from journals with only 1 publication.
That is also why we compute cs_{q3} were we only use the journals/conferences within the 75th percentile (Q3) based on the number of publications, which minimizes outlier influence of journals with a high number of unique authors and a low number of publications from within the journal of the given author. The impact of only using the third quantile can be seen in Table 2.2.

Journal	Publications	CS
ACM TG	5	2492
CGF	5	5793
TVCG	14	7608
cs_j	24	15 893

Table 2.2: Overview of an imaginary authors community size within Q3. Compared to Table 2.1, the cs_j of the author was reduced by over 75% by using the third quantile based on number of publications.

Another factor which has to be considered is that there are journals/conferences with a very wide range of topics which can inflate an authors community size. In the same

vein, publications to e.g. arXiv[1] must also be considered, as they are not peer-reviewed and one purpose of them is to reserve ideas for future publications. That is why the user can disable journals and conferences with a filter.

On its own, the community size of an author does not paint a concise picture. We debated about normalizing existing metrics with the community size but in the end we decided that its best use would be in conjunction with other metrics and visualizations.

2.2 Visual Comparison

Our second task was to create a platform to compare scientific authors. Part of the second task was to display and/or visualize established metrics, our new metric and the other data such as yearly trends, citation distribution and journal and conference distribution. First we will talk about yearly trends.

Publications per year is a metric which describes the author's output quantity over the span of his career. When comparing two authors and their output quantity, it can show that e.g. one author had his peak number of publications ten years ago and has significantly slowed down over the last five years whereas the other author slowly ramped up his output over the last ten years and has had significantly more publications within the last years. This can show the second author is more active within recent topics and more up to date on recent academic trends within his field.

Similarly to publications per year, **citations per year** represents how many citations he has received within the given year. Whereas publications per year represents quantity, citations per year can act as a measurement of quality, due to the common consensus that more citations correlate with a higher quality of publication. The contrast between quality and quantity is very important since it can show that an author has published a lot during a time period whilst also barely receiving any citations. It is therefore possible, that they received most of their total number of citations during a time where they barely published anything. It is important to note that a publications citations were not necessarily received during the year the publication was published in.

Publications per journal/conference can narrow down an authors field even more, e.g. from computer science to HCI or visual computing and additionally shows in which journal or conference the author primarily publishes in.

Author positions: In most scientific fields of study, the order in which the authors are listed on the paper indicate their amount of contribution to the paper. Other fields such as mathematics list their authors in alphabetical order. In this paper we will focus

[1] https://arxiv.org/

on fields which sort by contribution. Being listed first on a paper indicates that the author was the primary author of the paper, i.e. he did most of the work, whereas being listed between second and second to last indicates that he helped with some parts of, reviewed or proof-read the paper. The last position is usually the one who contributed the least, e.g. when a PhD student publishes a paper, the student is listed first, his PhD colleagues are listed in the middle and the student's boss (i.e. his professor) is last. The grouping of first, middle and last was adopted from Post et. al.[5] from their calculation of their new indices. This approach also makes sense from logical standpoint since for example, in a paper with eight authors, the difference between the 6th and 7th position is negligible.

Displaying the **top 5 publications** (sorted by number of citations) simply shows the standout papers of an author. It also shows the difference between peak performances. **Citation distribution** simply shows the Interquartile Range of the citation values. It helps the user to quickly analyze the distribution and shows whether the data is skewed or not.

3 Related Work

In this chapter, we will focus on some established alongside some lesser known metrics and tools which allow the user to take a look at an author's scientific merits.

3.1 Established Metrics

The **h-index**[4] (Hirsch index) is the highest number of publications from an author which were cited at least h times. It was developed to address flaws in other bibliometric metrics such as *total number of publications* or *total number of citations* which can be majorly affected by a single publication which does not accurately reflect the quality of an author. One of the issues of the h-index is that it does not consider the author's position in the author listings, which in some scientific fields is significant to show the author's contribution. For example, professors are often listed last, which usually indicates that they only contributed to a very small amount of to the publication. Another issue of the h-index is that it can be manipulated by self-citations or coercive citations in which an editor forces the author to cite his papers before he agrees to publish the author's publication[9].

The **g-index**[2] is defined as the unique largest number, such that the combination of the top g articles received at least g^2 citations. Compared to the h-index, the g-index averages the number of citations and does not ignore publications with the highest number of citations. Neither the h-index nor the g-index tell the full story of an author's accomplishments. But when taken together they present a more concise picture of an author's accomplishments[6]. It is especially useful in the case of new, highly cited publications since, for example, an h-index of five could mean a variety of things. It could represent a total of five publications with five citations each. It could also mean there were four highly cited publications (more than 200 citations each) and one publication with five citations[3]. Once a publication reaches the threshold to be included into the h-index, its subsequent citations no longer matter for the h-index.

The **impact factor**[1] of a journal for a given year y is defined as in Equation 3.1. Compared to the other metrics discussed in this paper, the IF (impact-factor) is not used to assess the output quality of a specific author, but to assess the importance of a given

[1] https://clarivate.com/webofsciencegroup/essays/impact-factor/

$$\mathrm{IF}_y := \frac{c_{y-1} + c_{y-2}}{p_{y-1} + p_{y-2}} \tag{3.1}$$

Figure 3.1: The definition of the impact factor with y being the year, c_y being the total number of citations from the year y and p_y being the total number of publications from the year y.

$$\text{i10-index} := \text{number of papers with 10 or more citations} \tag{3.2}$$

Figure 3.2: The definition of the i10-index. Google Scholar also displays the i10 index for based on the publications within the last five years.

journal. It reflects the average number of citations for a given year (and its previous year) published within the journal.

The **i10-index** was created by Google Scholar[2] and is used in the *My Citations* feature. It is defined as in 3.2. Only counting publications with 10 or more publication provides a quick look at the author's overall performance and is easily calculated but it does not consider the author's community size and average and median number of citations. Additionally, it is only used by Google Scholar.

c-index and Subindices of the h-index: New Variants of the h-index to Account for Variations in Author Contribution by Post et al.[5] suggested a variation of the h-index called the c-index, which includes the authors position in the author listing. The separated author positions into first, second, second-to-last and last, which we slightly modified and adopted into our display of author positions (see Motivation chapter). The c-index has enhanced recognition for the primary position (first, second) and senior position (second-to-last, last), which provides a more detailed view of an author's accomplishments and uses the authors h-core[6] articles, which is defined as the articles of an author with equal or more citations than his h-index. The main problem of this approach has is that not every scientific field orders their authors by amount of contribution.

The **m-index**[3] also known as the m-quotient, is defined by h/n, where h is the h-index and n is the number of years since the first published publication of the author, which takes the length of the author's career into account. As Hirsch noted[3], the first publication might not be always the best starting point, since it could have been only a minor contribution and the m-index can also have a negative impact on part-time academics or academics who had career interruptions (parental leave etc.).

[2] https://scholar.google.com

3.2 Exploration tools

In this section we will go over popular tools to explore scientific metadata. All of these tools lack the feature to compare authors, a feature which this application offers.

Google Scholar[3] is one of the more popular tools to access scientific metadata and is managed by Google. As previously mentioned, the i10-index was created by Google and is only used in Google Scholar. For author metadata it features a list of publications ordered by number of citations, the i10 and h-index (total and last five years), a bar chart displaying citations per year, a list of common co-authors and the author's field of study in the form of a list of keywords.

lens.org[4] is another tool to explore scientific metadata. For author metadata it displays a list of publications, a list of institutions the author has worked for (+ world map highlighting the countries), a publications per year bar chart, a bar chart for common co-authors and a bar chart for the authors most common keywords. It also has an analysis feature which displays a more detailed publications per year bar chart (stacked bar chart based on publication type), a word cloud for the author's keywords and many more graphs mostly related to citations and institutes.

sonne[7][5] is a tool which lets the user explore metadata in a graph like fashion. At its core, it allows the user to track connections between authors, publications and journals/-conferences. It features three datasets (which will be discussed in the next chapter) and allows the user to apply a large variety of filters to a search query. For author specific metadata, it shows a list of publications, an author position and publications per year bar chart and lists for publications per journal/conference. If the Semantic Scholar (s2) dataset is selected, it also features a list of keywords ordered by usage count.

[3] https://scholar.google.com
[4] https://lens.org
[5] https://sonne.0ds.de

4 Data Sources

There are several "free" data sets for publication metadata, each of them with their advantages and disadvantages. This chapter focuses on various providers, their features and their drawbacks. While the size of the data at hand was a large factor in determining which data sources to select, being freely licensed and having a good and easy way to acquire (mostly JSON or XML REST-API's) the data was also an important factor.

Microsoft Academic Graph[8] (licensed under ODC-BY 1.0) is the largest available database for publication metadata and is offered through Microsoft's Azure Cloud Computing platform[1]. Although Microsoft provides the database free of charge, storage, ingress and egress are subject to Azures pricing and requires a Microsoft account. The setup process is quite tedious and time consuming. The database is stored as TSV files and is separated to support relational databases and is updated every one or two weeks. It was built with Microsoft's Bing search engine crawlers and was post-processed with artificial intelligence[2]. While being the largest available database, it also requires the use of Microsoft's proprietary software and cloud platform. Luckily, Microsoft offers the option to download the entire database as TSV files, which was used to populate the database of Schmid[7].

CrossRef[3] offers an API to access metadata for over 90 million publications. While their documentation is excellent, there are some issues with CrossRef. Free access will be rate-limited once the user reaches a certain threshold and having rate-free access requires a monetary subscription[4]. Additionally, computing the total number of unique authors for a journal is highly complex and requires multiple API calls and even then does not have all the data we require for this application.

Semantic Scholar[5] is another public, free API for publication metadata (licensed under a custom non-commercial license). Compared to the other sources, Semantic Scholar is by far the smallest available database and also has one of the highest rate-limiting's (100 requests/5 minutes), which is unusable for our use-case. They also provide keywords for the publications, which is a rare feature for publication metadata databases. Additionally,

[1] https://azure.microsoft.com/
[2] https://www.microsoft.com/en-us/research/project/academic/
[3] https://github.com/CrossRef/rest-api-doc
[4] https://www.crossref.org/services/metadata-delivery/plus-service/
[5] https://api.semanticscholar.org/

they use their own ID system for authors and publications and their own metrics to determine whether a publication counts as 'influential', which means we would have to adopt these systems.

Google Scholar: Another popular site for accessing publication and author metadata is Google Scholar. While being arguably one of the most used sites, Google simply restricts automatic scrapping of their content and they do not provide a public API which immediately eliminated them as a possible data source.

IEEEX is another possible source, they provide an API that requires an account on their platform, however their database is small compared to the other sources. They are also severely rate-limited and their platform had, at the time of writing, several technical issues (e.g. expired SSL certificates).

Almost all of the aforementioned databases do not provide keywords for publications, which could be used to show an author's participation in recent scientific trends such as machine learning, blockchain, etc. The only platform which stores keywords is IEEEX and Semantic Scholar but their drawbacks, as explained above, makes them unfit for our purposes. For future work, one might implement the acquisition of keywords either through Schmid's[7] backend (since he also indexed Semantic Scholar) or through other services. As a result, we used Schmid's[7] backend with MAG dataset to acquire the required data, since his work needed a similar data layout. His database is built upon Apache Solr[6] and provides either web-socket based or HTTP based communication and is hosted on bwCloud[7].

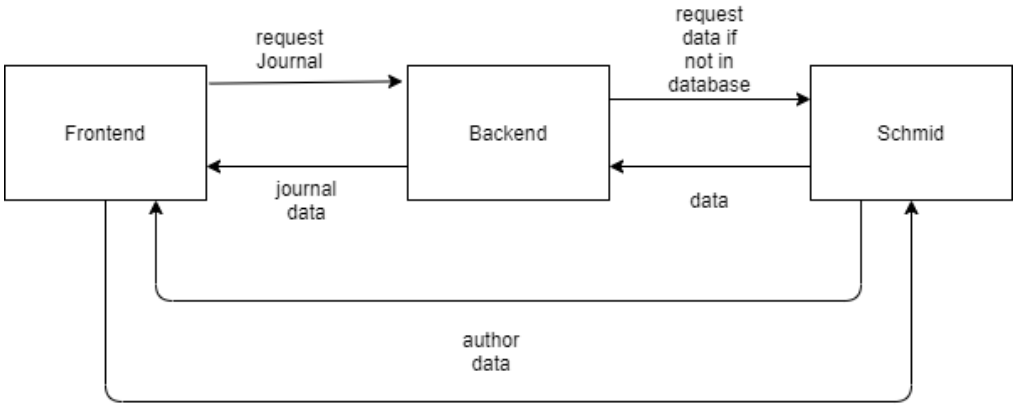
[6] https://lucene.apache.org/solr/
[7] https://www.bw-cloud.org/

5 Implementation

The application is divided into two parts, a web-frontend and backend. The backend handles the data acquisition and storage, whereas the frontend displays the data in meaningful ways. The frontend was developed as a website due to the widespread access to browsers on every major platform (i.e. even mobile, although this application is not optimized for mobile devices). This decision also avoids having to deal with platform specific limitations and issues. The backend is a simple REST-API written with the popular language Golang[1] due to its simple tooling. The details of these components will be discussed in the upcoming sections.

5.1 Architecture

Figure 5.1: Workflow of the application. The frontend requests the author data (a list of publications) directly from Schmid's server[7]. Then it builds a unique list of journals and conferences and requests the community sizes for each. The backend will return the data if it is present in the database or request it from Schmid's server, store it in the database and then return it.

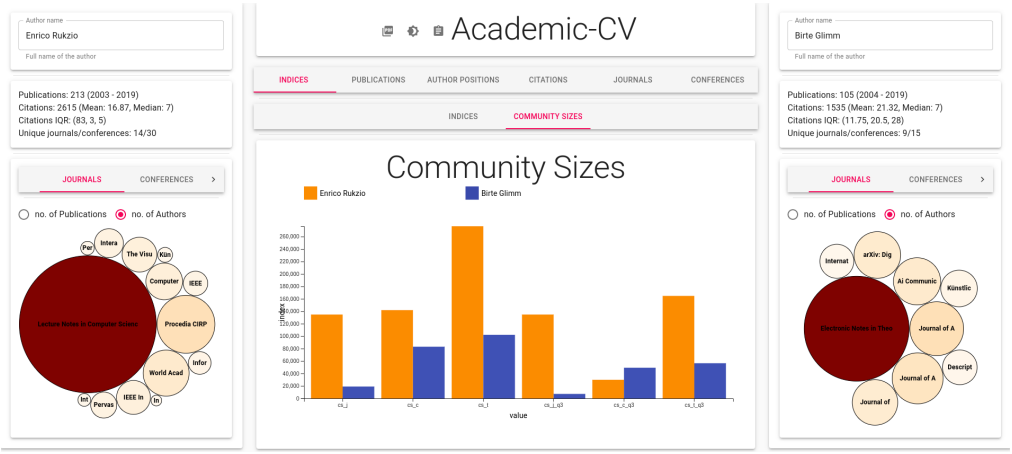


The frontend can be divided into two main parts, the details for a given author on each side and the comparison between the authors in the middle (see Figure 5.2).

[1] https://golang.org/

The author-details-component (see Figure 5.3) fetches the author data from Schmid's server[7], which includes a list of the author's publications. For each publication the data contains the title of the publication, citation count, journal or conference (if applicable) and all contributing authors.

Figure 5.2: An overview of the frontend with the two author-details components on the sides and the comparison in the center. The UI was designed for desktop use and Full-HD resolution and is therefore not optimized for mobile use.



Once the request returns, the data is processed and values such as the h-index or g-index are computed. After that, the data for each unique journal and conference is requested from the backend. If the data is present in the database, it will just return the stored information, if not, the data will be fetched from Schmid's server[7], saved to the database and then will be returned to the frontend. After all those requests are finished (which can take quite a lot of time), the community size is computed.

The bubbleplots display the journal/conference data. The radius-value can be either the number of publications the author has published within the journal/conference, or the community size of the given journal/conference, which provides an overview of the distribution of the author's community size. The radius-value can be toggled with a radio button above the bubbleplot. Additionally, the author-details-component shows a list of the author's publication sorted by publication date (newest to oldest).

The comparison between the authors (see Figure 5.2), in the center of the application features three utility functions in the title bar. A link to this paper, a way to toggle between light and dark mode (preference is saved to localStorage) and a button to generate a link to the current comparison, which is automatically copied to the users clipboard. The main part consists of a tabbed user interface, with each tab showing a different metric-comparison between the authors. The current tab-index is also saved to localStorage. Each component for comparisons is written as generically as possible. For example,

Figure 5.3: The author-details component displays the total number of publications by the author, the years the author was active in, the total number of citations he received with the mean and median also provided and the total number of unique journals and conferences the author has published in.

the bar charts are written to receive two arrays of data, one for each author, with each element having two fields (x-value and y-value). To access these fields, the bar chart also receives two anonymous lambda functions. This way, the charts can easily be reused for different use-cases in future iterations of this application. The data is already in the proper shape due to the processing done in the author-details-component, which means that these computations are done once the data is loaded and are then simply passed to the components.

5.2 Tech-Stack

This section explains how and why we used the frameworks to build the application. To enable easy reproduction and potentially enable open sourcing the code for this application in the future, most of the frameworks we used are open source.

5.2.1 Frontend

The web-frontend was written with the popular JavaScript library **ReactJS**[2] developed by Facebook. Its component based approach and built-in lifecycle hooks make it easy to iteratively develop an interactive user-interface. Each component has its own internal state and automatically re-renders its view based on state changes (user-input, returning AJAX-calls). Additionally, each component provides functions for its lifecycle such as when the component is mounted or unmounted to the DOM. This provides finer control over the application's flow and makes it easier to deal with side-effects such as dealing with asynchronous tasks. The component based approach also enables the developer to create simple, generic and reusable components to be used within the application, i.e. writing a list which can render all sorts of containers. For example, the list can be written to accept a simple array of objects and an anonymous function to access the desired key of the object or just an array of strings, which makes the list widely reusable. One issue ReactJS has, is that while passing data down the component hierarchy is very easy and straightforward, passing data up the hierarchy can be quite tedious and complex, as it requires the use of DOM references and many callbacks. The solution to this issue will be discussed in the mobx section. ReactJS was chosen due to the author's prior experience with the library, the ability to quickly prototype components and the extendibility and compatibility with other libraries/frameworks. Other options like Vue.js[3] or Angular[4] were considered but were ultimately dropped due to tooling issues or subjective preferences.

mobx[5] is JavaScript state-management library. It can be used to manage the internal state of the application and its ReactJS plugin allows state changes to automatically trigger re-renders. Its key principle consists of state variables that are wrapped within an observable. Changes to observables are tracked and trigger certain reactions such as re-renders and the change of auto-computed values. mobx also enables the developer to create a global store within their application, which can be accessed from anywhere, which eliminates the passing-up data problem of ReactJS as discussed in the ReactJS section. It also provides options to react to certain observable changes

[2] https://reactjs.org/
[3] https://vuejs.org/
[4] https://angular.io/
[5] https://mobx.js.org/

without having to check in intervals whether the operation is already done, e.g. the state of asynchronous I/O operations can be bound to observable values which do not require constant tracking.

Material-UI[6] is an open-source implementation of Google's Material Design[7] in ReactJS. It offers a wide variety of pre-styled ReactJS-components, which lowers the amount of required CSS by a significant amount. As a result, the application features less than 20 lines of CSS code. The built-in layout components also significantly lower the amount of troubles caused by alignment issues in raw HTML5+CSS. Since it follows the Material Design specification, the application has a unified and familiar look without having to write custom stylesheets and looks great out-of-the-box. It also has support for global themes, which allows the developer to quickly change the look the application without having to rewrite everything.

D3.js[8][1] is a data driven JavaScript library, which binds the data to the DOM using HTML, CSS and SVG. It was used to create charts and graphs used within the application. It also offers a wide variety of math utility functions (e.g. computing mean/median values, building quantiles etc.), which, for example, were used to compute the community sizes. As one can imagine, writing a dynamic tree map from scratch can be quite difficult. With D3.js however, it only took 100 lines of code.

5.2.2 Backend

Due to the complexity of our requests, the results of the queries for a number of unique authors for journals and conferences are stored in a MariaDB-database[9]. The web server was written in Go[10] with the libraries gorm[11] (database abstraction based on structs) and gorilla/mux[12] (http routing). The server simply checks if a given journal/conference is already stored in the database. If that is not the case, the server requests the data from Schmid's[7] server, saves it to the database and then returns it as a JSON-response. If it is stored, it simply returns the data in a JSON format. Go was chosen due to the author's familiarity with it, its type-safety, speed and most importantly its single-binary output on builds. The binaries are quite large in file size (~10MB for this application) compared to other languages, but all dependencies are bundled within, which makes deployment extremely easy and avoids having to deal with complex build processes (e.g. C++/CMake or Java/Maven). Another great feature of Go is the support

[6] https://material-ui.com/
[7] https://material.io/
[8] https://d3js.org/
[9] https://mariadb.org/
[10] https://golang.org/
[11] https://gorm.io/
[12] https://github.com/gorilla/mux

for pointers and references while simultaneously being a garbage-collected language, which greatly eases up the development process. The web-server is less than 200 lines of code and the database is also very simple, since it only has two tables with two columns each (ORM columns omitted). A big advantage of using an ORM is the ability to use an object-oriented approach (i.e. the database returns objects/structs instead of single fields) and a specific advantage to GORM is that it automatically inserts an ID system and generates columns for created, updated and deleted timestamps within the table. Due to the backends low complexity, any programming language with database drivers and web-server capabilities would have worked. Golang was chosen due to its performance (which is negligible in our case), its great tooling (dependency management, integrated code-style etc.) and mostly personal preference.

6 Visualization

Each type of visualization described in this chapter was rendered via the JavaScript library D3.js[1]. The following chapter will go over every metric we discussed in Chapter 2, how it was visualized and what information can gathered from the visualization.

Metrics & Numbers: Established metrics such as h-index or i10-index are displayed via plain numbers. Generally, graphs are usually easier to read and interpret thanks to the visual aspect of them. Visualizing those values with a graph can waste a lot space since for example, a bar chart with only two values on the x-axis (left and right author) barely provides any significant advantages over just plain numbers. Alternatively, combining the metrics into one bar chart can cause readability issues due to the metrics potentially having a different range of y-values. One metric could have a range from 0-100 and the other one a range from 0-5000, which can also lead to false conclusions by not accurately showing the difference in the first metric between the two authors since for example the difference between 10 and 30 looks insignificant in a bar chart when the y-axis has a maximum of 5000. Other solutions such as stacked bar charts or bar charts with multiple y-axes could have been used (and could potentially be implemented in future improvements of this application) but for the time being we chose to display those metrics in a table. Values such as the median or mean number of citations of an author could also have been included within the bar chart for citations per year as a line, but due to their commonly low value of them we elected to just display them in the author-details component and in the legend for the charts since a line which barely hovers over the x-axis is quite hard to read and interpret.

Yearly Trends: Another goal was to visualize the author's number of publications and citations over time to show how the author has performed over the span of his career. Both of those metrics consist of a two dimensional data set with a categorical value (year, x-axis) and a numeric value (number of publications/citations, y-axis). That is why we chose to visualize them with bar charts. To properly visualize those metrics, we first have to compute the maximum value of the y-axis and build the range of the x-axis by using the 'years active' value from the previous talking point. Since we are using two datasets per chart (left and right author), we first have to join the datasets and then build the axes. This is also how we generate one of our regular and grouped bar charts,

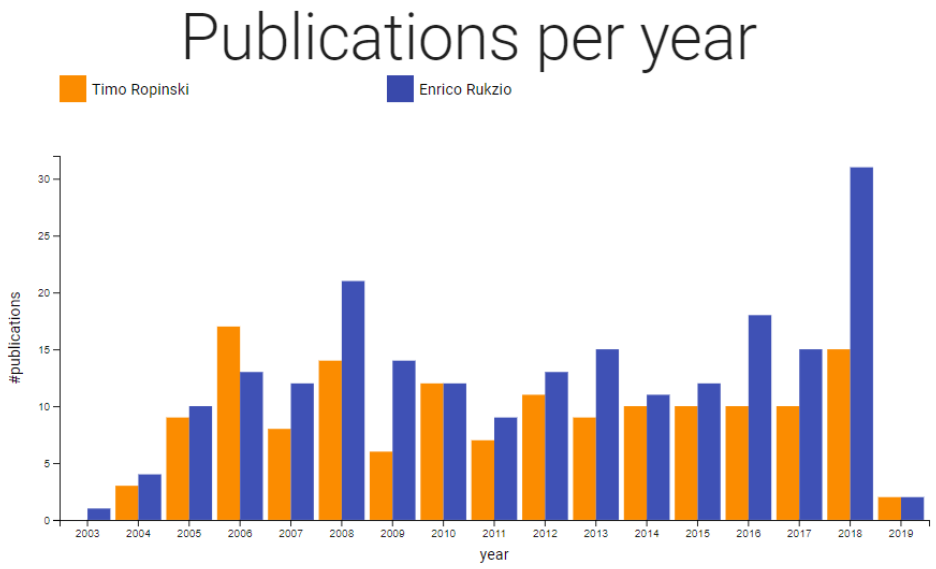
since we display two values on the y-axis per one value on the x-axis. To differentiate between the left and the right author, we color coded the bars blue and orange, two easily distinguishable colors. An example for such a bar chart can be seen in Figure 6.1. Since we apply post-processing to almost all of the data we receive, the data for those metrics is already in a proper shape once the applications has finished loading. A simplified example of this can be seen in the Listing 6.1
Now that we have discussed how these metrics are visualized, we need to discuss why we visualized them and what kind of information can be gained from those charts.

```
const data = [
   { year : 2008, publications : 69 },
   { year : 2009, publications : 420 }
];
const xAccessor = (d) => d.year;
const yAccessor = (d) => d.publications;
```

Listing 6.1: A simplified data shape for bar charts. Due to JavaScripts dynamic typing system we can store the accessor functions in variables.

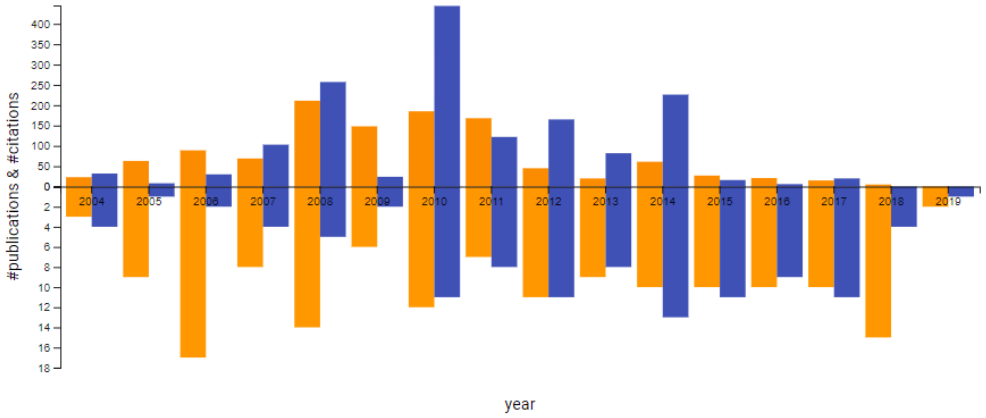
Publications per year is a metric which shows how many publications an author has published in each year of his active career. This allows the user to easily take in information. If e.g. one author has primarily published at the start of their career and has significantly slowed down over the last five years or if they gradually increased their output quantity over the span of their career. Combining this metric with others (see Figure 6.2) gives us a good overview of an author's output quality. If the author for example has very few publications at the start of their career and a lot within the last five years, but their number of citations is a lot higher at the start of their career compared to recent years, we can conclude that the quality and/or significance of his recent work is lower than their early work. Adding the h-index to the picture would then also show that their h-index primarily comes from their early work and would probably be a lot lower if only their recent publications were used for the calculation of their h-index since the h-index does not consider the publication date. When comparing two authors, publications per year also shows which one of the authors is probably more up to date on recent topics, which can be quite important based on what this application is used for (e.g. hiring process).

Figure 6.1: Example of a bar chart showing the publications per year metric. The x-axis shows the year whereas the y-axis shows the number of publications each author has published within the year.



Citations per year is a similar metric to publications per year. Instead of the y-value being the number of publications published during the year it is the number of citations the author received for all his publications during the year. As with publications per year, it is best used alongside other metrics. For example, if the value of the left author is on average only slightly higher than the one of the right author but the overall community size of the right author is significantly smaller than that of the left author, we can conclude that the difference between the two authors cannot be taken as is due to the left authors number of citations being automatically higher due to his bigger community and therefore bigger impact factor of his journals/conferences. Since the citations per year metric has the same x-axis as the publications per year metric we also implemented a mirrored bar chart (see Figure 6.2), where one can easily see if an author's number of citations comes from a time where the author published a lot or not. It is important to note that the citations do not necessarily originate from the year that the specific paper was published in, but since the work for the citations was done in that year we attribute them to the release year of the publication. Additionally, if for example, the left author has far more publications in a certain year than the right author but far fewer citations, we can conclude that the quality of the left author's publications in that year is far lower than the that of the right one.

Figure 6.2: Publications/Citations per year visualized in one graph. The number of citations is displayed in the upper part of the chart whereas the number of publications is displayed in the lower part. This chart makes it easy to see a correlation between the amount of output and number of citations. It is important to note that the citations do not necessarily originate from the year that the specific paper was published in.



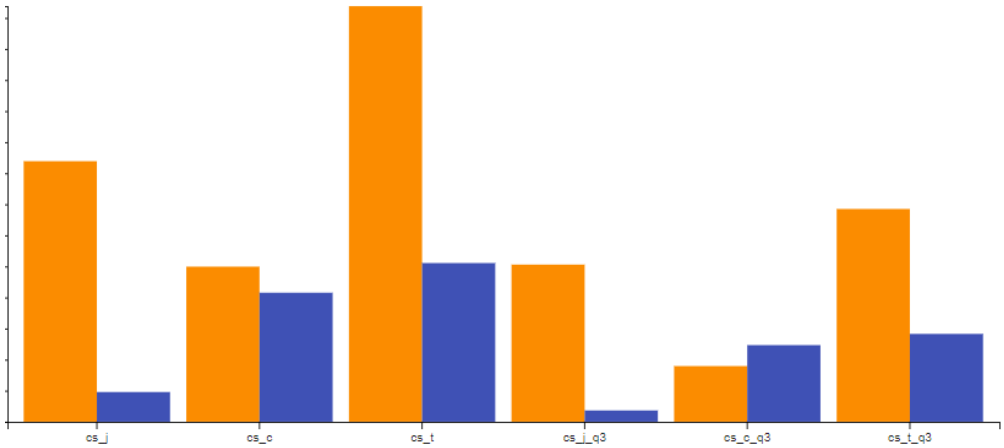
Community Size: The community size is a two-part metric. The first part being the values without any filtering and the second part having only the values that are within Q3 of the IQR. The x-values are simply the six different values (i.e. cs_j, cs_c etc.), and the y-values are the actual values of the sub-values. This gives an overview of how the community sizes between the authors differ and also shows if an author primarily publishes in journals or conferences. As seen in Table 6.1, those numbers do not paint a concise and easy to interpret picture when only looked at as numbers. That is why we visualize them via bar-charts as seen in Figure 6.3. With the bar chart, it is easy to see that the first author (orange) has a larger community size than the second author (blue). It also shows that the orange author primarily publishes in journals, whereas the blue author primarily publishes in conferences. The community size alone cannot be used to compare two authors. It is best used with other metrics such as citations per year or author position, both of which will be discussed within the following sections.

Author	cs_j	cs_c	cs_t
Author 1	167 687	99 822	267 509
Author 2	19 093	83 038	102 131

Author	cs_{jq3}	cs_{cq3}	cs_{tq3}
Author 1	101 137	35 807	136 944
Author 2	7 236	49 287	56 523

Table 6.1: Comparison of cs and cs_{q3} between two authors. Raw values on the left and values within the third quantile on the right. Numbers were taken from a real life example.

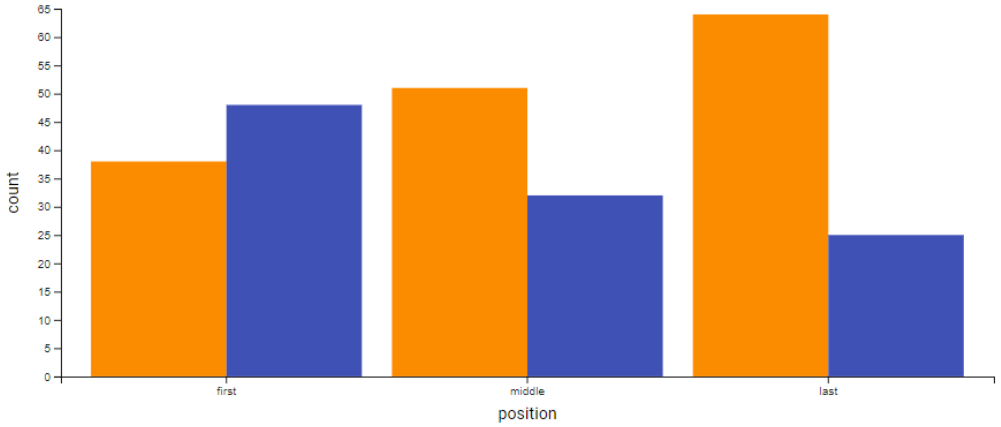
Figure 6.3: Numbers from Table 6.1 visualized as bar chart. As we mentioned before, a graph makes it generally easier to interpret those numbers, especially when comparing two sets of numbers.



Author position: The author position gets visualized in two ways. The first way is by simply taking the position as the x-value and the number of times the author was in that position as the y-value. The second way is grouping the position into first, middle and last. This approach was adapted from Post et al.[5], where they use it to calculate various sub-indices of the h-index whilst considering the author's position in the author listing, as being first usually indicates that the author has done the most work, middle indicating that author has helped the first author and last indicating that the author did some some very minor work (proof-reading etc.). Looking at Figure 6.4, the orange author was in the last position for most of his publications, the second most being middle and the least being first, whereas the blue author has the exact reverse distribution being first the most and being last the least. This indicates that the orange author has done far less work for his publications than the blue author. In a university environment, this could mean that the orange author is the a professor and most of his publications were written by his PhD-students. It is important to consider that not every scientific branch orders their author listing by amount of contribution, e.g. mathematicians usually order alphabetically.

Another goal was to show in which journals or conferences the author mostly publishes in. Since this is another categorical dataset we could have used bar charts to visualize these metrics, but due to certain journals/conferences having very long names, we selected tree maps for the task. Tree maps are usually used to display hierarchical data but in our case the data is strictly one dimensional. Alternatively, we could have also used pie charts but similarly to bar charts the long names caused readability issues.

Figure 6.4: Example for grouped author positions. In this case the left author (orange) has done less overall work for his publications than the right author (blue).

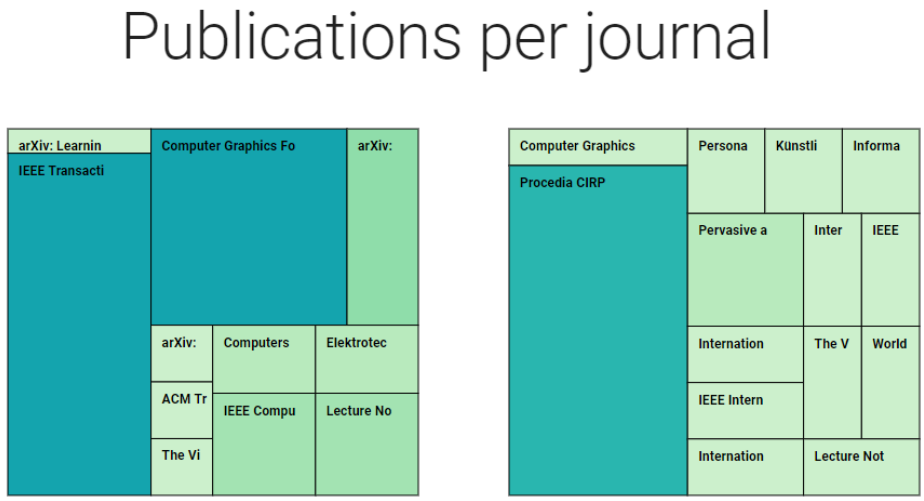


Another option would have been bubbleplots, but since we are comparing two datasets side-by-side, the equal shape makes it easier to read. The tiles of the tree maps were color coded based on the number of publications the author has published within the journal/conference. An example for such a tree map can be seen in Figure 6.5.

Publications per journal/conference: This visualization gives an overview in what subcategory of their field they publish in the most, e.g. visual computing, artificial intelligence, human-computer-interaction etc. Having two tree maps side-by-side allows the user to easily identify the primary focus of the two authors and makes it easy identify their primary fields of study. Additionally, this makes it easy to identify if the author has published a lot in either very broad topic journals (e.g. Lecture Notes in Computer Science) or not as highly regarded ones such as arXiv due to the lack of peer reviewing. If the user hovers over one of the boxes in the tree map, the corresponding journals/conferences full name, number of publications from the author and community size are shown in a tooltip.

A variant of this metric is also present in the author-details component as a bubbleplot as seen in Figure 6.6. The difference is that the bubbleplot is part of the UI and can be interacted with. The radius value can be toggled between the number of publications the author has published in that journal/conference or the community size. By toggling the value or by using the bubbleplot with the tree maps, one can easily see the community size of the authors primary journals/conferences. Additionally, the bubbles can be used to filter out certain journals/conferences for the community size computations by clicking on them. Once a bubble is disabled, it is greyed out and the text is crossed-out. This way the user can easily filter out journals such as arXiv, or irrelevant topics in an application

Figure 6.5: Example of a tree map showing author-journal distribution. One can easily see that the left author's primary target journals are CGF and TVCG (IEEE Transactions. . .) whereas the right one's are Procedia CIRP.

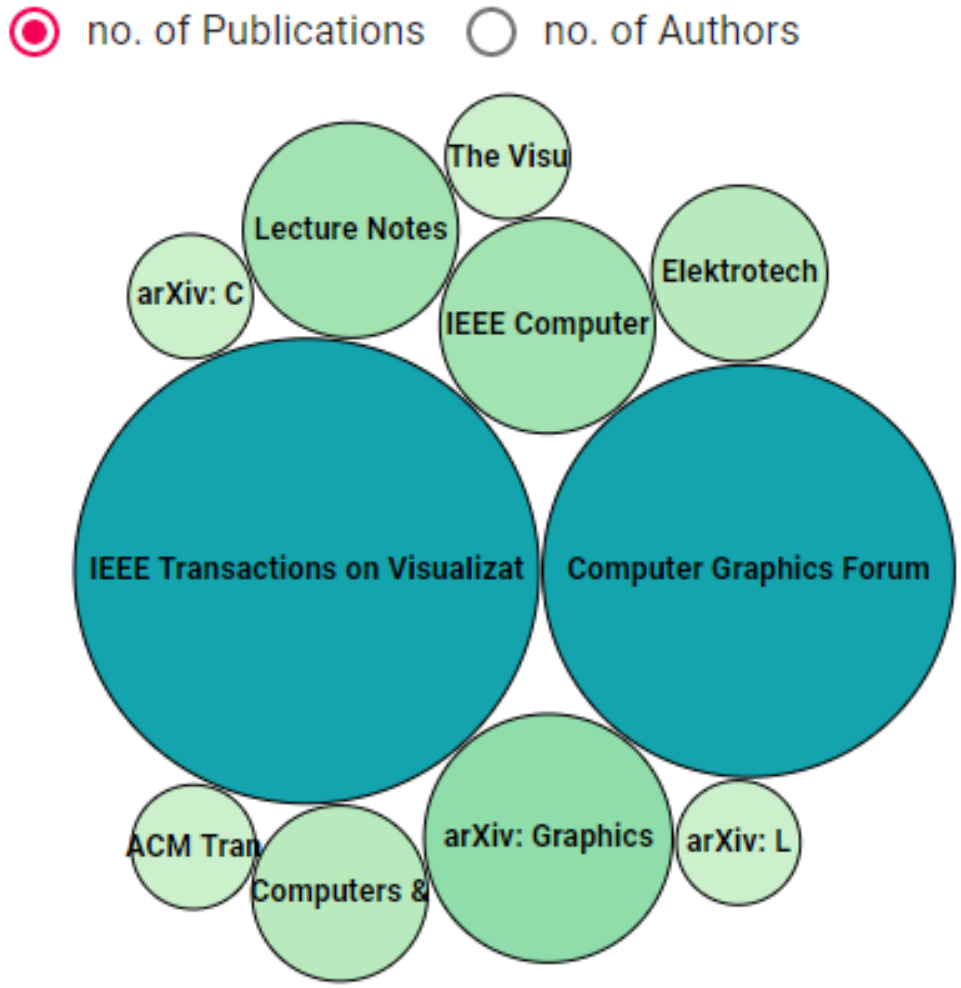


process scenario. Similarly to the tree maps, each bubble has a tooltip which shows all values (name, community size and number of publications) on hover.

Citation distribution: Showing the citation distribution with a boxplot allows the user to easily identify the median performance of the author's publications and whether or not the author has a lot of outlier publications, i.e. publications outside of the IQR, which means outside of the standard deviations.

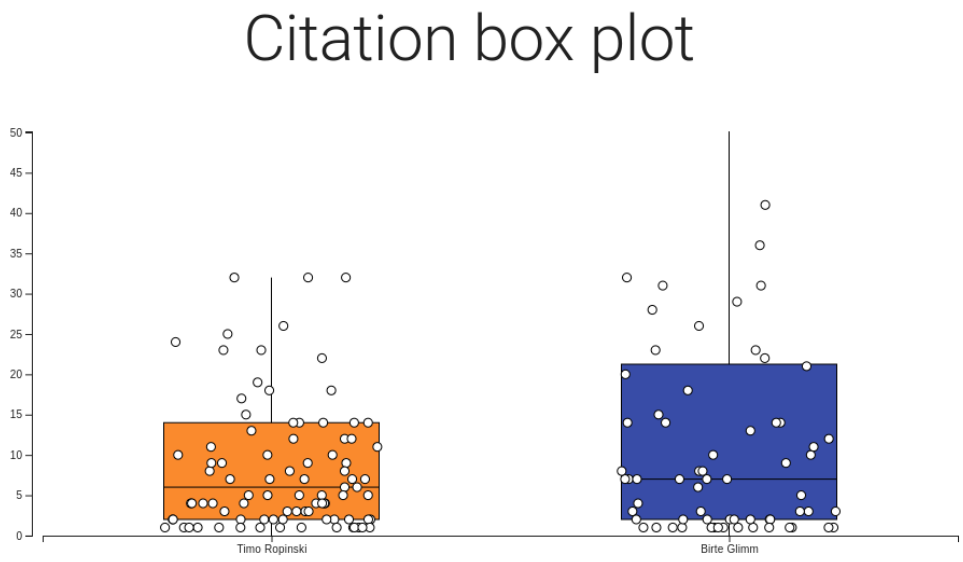
Boxplots are a graphical way to visualize a distribution of a certain attribute. In this case, the interquartile range (IQR) of an author's number of citations. The horizontal line within the rectangle is the median value, the upper line is the Q3 (upper 75%) and the lower one is the Q1 (lower 25%). The vertical line displays the author's min/max value of the IQR, with min being defined as $q1 - 1.5 \cdot \text{IQR}$ and max being defined as $q1 + 1.5 \cdot \text{IQR}$. Every point within the rectangle is within Q2 (median). The IQR is a great way to detect outliers and gives a good overview of the general distribution. To avoid gross outliers and very unreadable boxplots, we chose to filter out everything above the maximum by default. A checkbox toggle allows the user to disable that filter. Additionally, each data point displays the title of publication and its number of citations in a tooltip on hover.

Figure 6.6: Example of a bubbleplot showing journal community sizes. The toggle buttons at the top allow the user to toggle the radius between the community size (no. of authors) and the number of publications.



Lists: Lists are just a series of items, most commonly displayed in a vertical fashion. There are two lists within the application. The first one is in the author-details component which shows a list of all publications of the author ordered from newest to oldest. If the database included a DOI (Document Object Identifier), a button that redirects to the DOI is present on the list item. Each list item displays the title of the publication as the primary text and, if present, the publication year and number of citations as secondary

Figure 6.7: Example of a boxplot showing the citation distribution. Every data point above the maximum ($q3 + (1.5 \cdot \text{IQR})$ is not shown. The vertical line displays the minimum/maximum for each author, the box is the authors Q2 (upper line = Q3, lower line = Q1) and the horizontal line within the box represents the median.

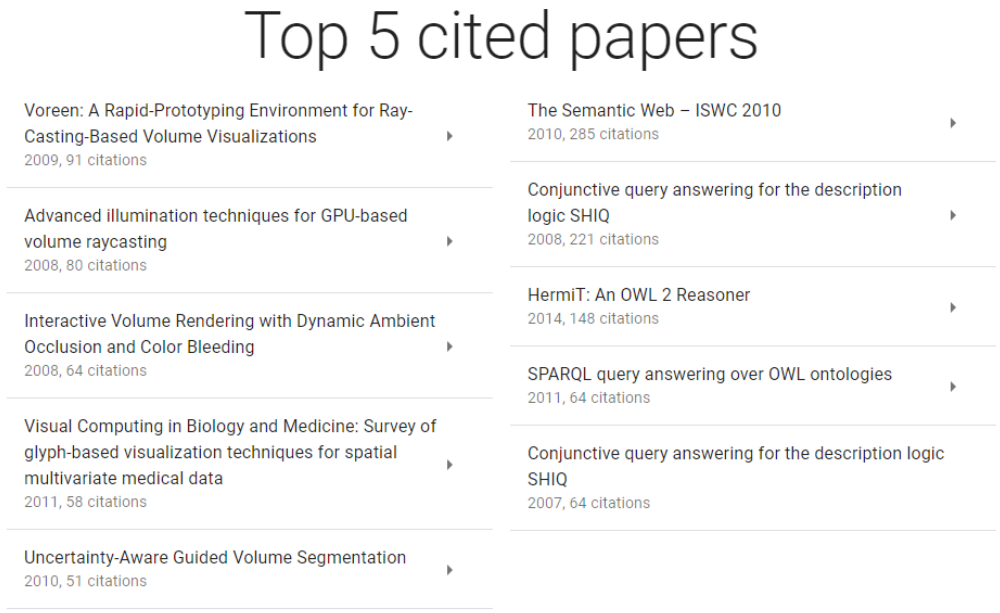


text. The second list is in the comparison component and shows two lists side-by-side with the top 5 publications (as seen in Figure6.8) of each measured by the number of citations they receive. This allows the user to easily inspect an author's best work and stands in contrast to the boxplot, since it only shows outliers. Additionally, in the journals/conferences section of the comparison component, there is a list of the author's journals/conferences sorted by number of publications.

For easy navigation, the frontend features a tabbed UI separated into the following categories:

- Indices (established metrics and community size)
- Publications (per year, publications + citations per year, top 5 cited papers)
- Author Position (ungrouped, grouped)
- Journals (publications per journal, journal list)
- Conferences (publications per conference, conference list)

Figure 6.8: Example of the top 5 publications of the two authors visualized as a list. The arrow button on the right of each list item will redirect to the publication's DOI.



7 Case Study

Since talking about statistical metrics without examples does not provide a lot of value, this paper will compare the metrics described in the next chapter with two explicit people. Both are computer science professors at Ulm University. For the sake of anonymity, we will not mention their names. We will use the name 'Foo' for the first author, 'Bar' for the second. The order we go in is the same one a user would experience when using the application. We omitted the sections about journals/conferences (i.e. publications per journal/conference), since those parts require knowledge about the specific field of study. A note for the reader: the dataset we used sometimes does not have all the information for every publication (e.g. citation count). Additionally, when we say an author's metric is better than the other, we purely base that on the numbers we have available. One cannot definitively say one author is better than the other, since some people may value quantity over quality, while others may value a high h-index.

As seen in Table 7.1, despite Bar's career is only one year longer, he has almost twice the amount of publications and more than 1000 citations more compared to Foo. These metrics alone generally speak in favor of Bar but as we mentioned before, most metrics alone do not paint a concise picture for the assessment of an author's performance.

Author	Publications	Career	Citations
Foo	105	2004-2019	1535
Bar	213	2003-2019	2615

Table 7.1: Overview of the authors' core data. Although Bar's career is only one year longer, he has more than double the amount publications and over 1000 more citations.

If we include the established metrics from Table 7.2, we can see that the i10-index difference is almost proportional to the difference in the number of publications. This can be explained by the mean citation count (Table 7.3) of both authors, where both have a mean citation count of over 10. The proportional difference between the h-index and g-index (author A has roughly 70% of Bar in both metrics) also shows that none of the authors have extreme outlier publications in terms of citation count.

Author	h-index	i-10	g-index
Foo	20	31	39
Bar	26	64	51

Table 7.2: Overview of the authors' metrics. The i10-index difference can be explained by the amount of publications and mean citation count of over 10. The non-proportional difference in the h-index can be explained by the higher mean citation count by Foo. (see Table 7.3)

Considering the mean and median citation count, the higher mean value of Foo and the equal median value (see Table 7.3) means that Foo's top cited publications have more citations than those of Bar due to the mean values being susceptible to outliers, which is confirmed by the actual top 5 publications of both authors as seen in Figure 7.3.

Author	Mean Citations	Median Citations
Foo	21.32	7
Bar	16.9	7

Table 7.3: Mean/Median Citations of both authors. The higher mean citations of Foo speak in favor for her work. The equal median mitigates that difference slightly due to the means susceptibility to outliers.

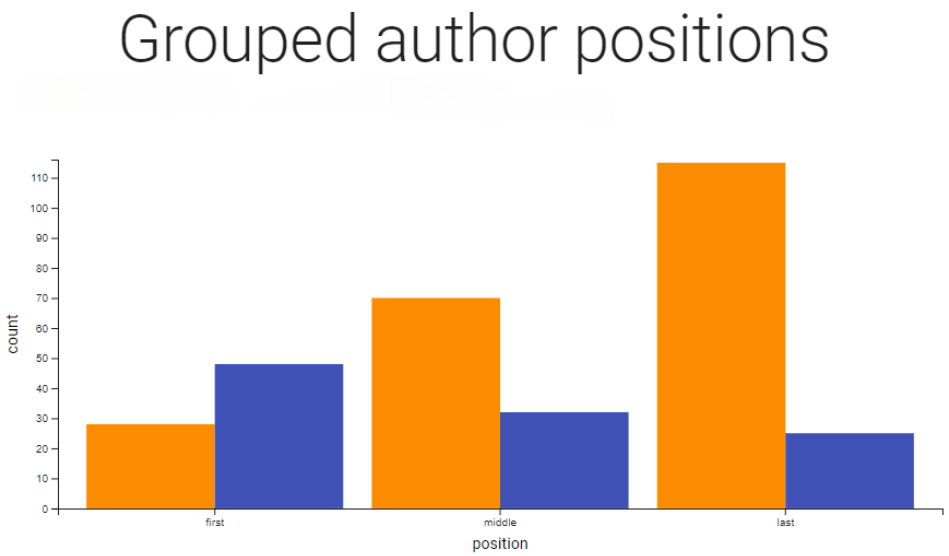
Since both authors are computer scientists, we can assume that the order of authors in the listings of their publications is sorted by the amount of contribution. Due to Foo's lower median and mean author position, we can assume that Foo has done more work for his publications. If we consider the distribution of the author position based on being first, middle or last in the listing (see Figure 7.1), the descending staircase shape of Foo is generally preferable. A ascending shape generally means that the author has not done most of the work for a large part of his publications.

Author	Mean Author Position	Median Author Position
Foo	2.41	2
Bar	3.6	3

Table 7.4: Mean/Median author positions of both authors. A lower mean and median author position can generally be considered better than a higher one.

Considering that author A has roughly a third of the community size of Bar (Table 7.5), both raw and Q3, combined with the aforementioned lower author position generally

Figure 7.1: The grouped author positions of both authors. Foo in blue, Bar in orange. Foo has a ascending staircase shape whereas Bar has a descending one. In general, a ascending shape means that the authors has done more work for his publications.



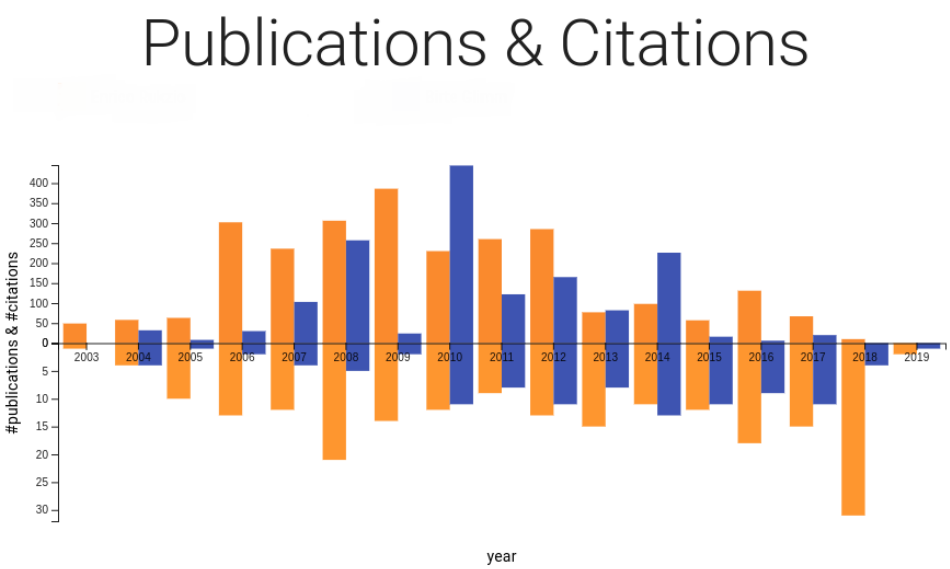
speaks in favor of author A since a bigger community size usually means a automatically higher citation count and h-index due to the impact factor of bigger journals, which is directly influenced by the number of publications published within the journal.

Author	cs_j	$cs_{j_{q3}}$	cs_c	$cs_{c_{q3}}$	cs_t	$cs_{t_{q3}}$
Foo	19093	7236	83038	48287	102131	56532
Bar	134734	134734	141823	29921	276557	164655

Table 7.5: Community sizes of the two authors. As shown in the last two columns, the difference between the raw values and the Q3 values stays the same with Foo having roughly a third of the community size of Bar.

By taking a look at the authors' yearly trends in Figure 7.2, we can see that both authors had their peak output to citations ratio between 2006 and 2014. The lower citation count for both authors from 2015 onwards can be attributed to either the dataset not having the necessary citation data or to the publications just being newer. Newer publications are generally not cited as often for multiple reasons such as new topics which are not as relevant yet or them having not as many reviews.

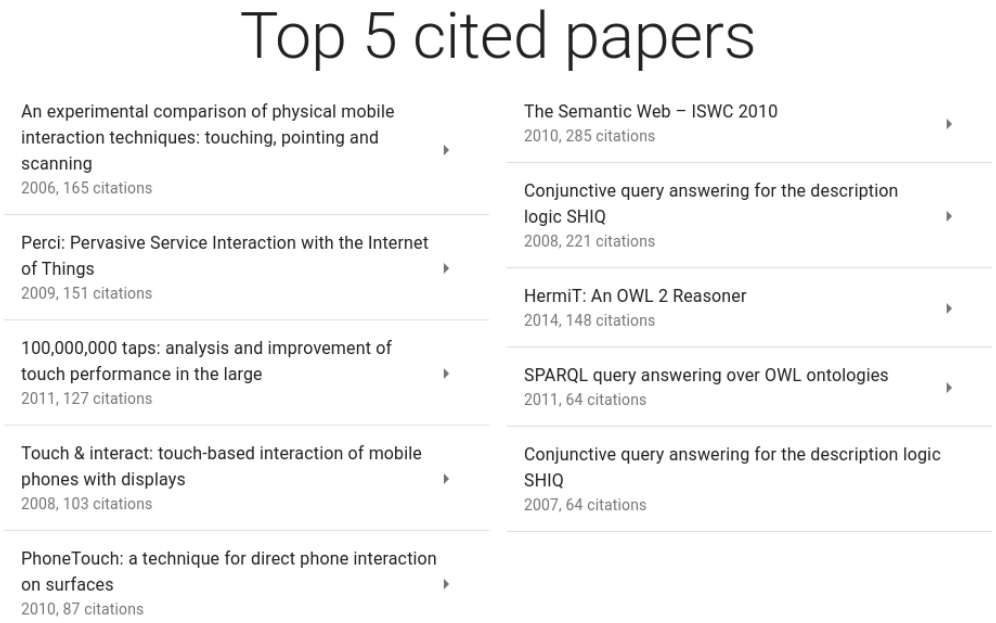
Figure 7.2: Publications/Citations per year of both authors. As seen in the chart, the output to number of citations received relationship was best for both authors around 2010.



In Figure 7.3, we can see that the most cited paper for Foo is publication in proceedings of a conference, which are generally higher cited than standalone publications since they contain multiple papers. Even if we ignore that, the second most cited paper of Foo has still more citations than the top cited paper of Bar.

Even after all that, we still can not make a conclusive decision on which author can be considered better or more effective since that would require an even more detailed look at the authors' careers and even more importantly, different people value different metrics. We simply presented and described the values we computed and visualized and valued them by our preferences, e.g. a lower author position being considered better.

Figure 7.3: Top 5 publications of both authors sorted by citation count. Bar on the left, Foo on the right.



8 Future Work and Conclusion

In this chapter we will go over our plans to further improve and extend this application and we present conclusion to this thesis.

Metrics: Adding more author-level metrics allows for a better understanding of an author's performance. At the time of writing, there are over 15 author-level metrics listed on Wikipedia[1]. Since most of them are not very established within the scientific community, we would have to add explanations to the UI explaining how these metrics are calculated. Most of the lesser known metrics try to include other factors such as author position, number of co-authors or the length of the author's career into them. This an attempt to mitigate the problems of the h-index discussed in Chapter 3. Another idea to further improve this application would be to create variants of the established metrics based on the author's community size, e.g. trying to normalize the h-index based on the difference in community sizes between the two authors. This would require a lot of statistical knowledge and research and is best left to people who have a far better understanding of the topic, but at the time of writing, we are not entirely sure if this approach would create great results and for the time being we think the community size is a metric which is best used when combined with other metrics such as publications/citations per year.

Caching and Updates: As discussed in Chapter 5, the list of publications is directly requested from Schmid's server[7]. An improvement would be to cache those lists in the backend database to reduce the load on the server. Another feature would be automatically update cached community sizes for journals/conferences after a certain amount time and give the user to the option to manually trigger an update. Since the current dataset we are using is not continuously updated, we chose to leave this feature for future work to avoid unnecessary overhead.

More Data: Obviously, access to more data means that we can visualize even more data. Right now, the next dataset to visualize would be the author's keywords to narrow down his field of study (e.g. Volume Rendering for Visual Computing, Virtual Reality for Human-Computer-Interaction) even further. The required data is already present on Schmid's server[7] but since we selected the MAG dataset and the keywords are only present within the Semantic Scholar dataset we chose to omit this feature for the time being to avoid having to deal with inconsistencies between the datasets. This feature

[1] `https://en.wikipedia.org/wiki/Author-level_metrics`

would also allow the user to easily identify if an applicant has the right focus on the desired topics without having to explicitly to look at the author's publications. We also thought about including more information about the author's career such as employment history but that kind of information is not properly available at the time writing and would require scraping the internet with web crawler, which can lead to licensing issues and potential IP blocks. One of our initial goals was to include a Venn diagram for an author's community size but due to the extremely high computation time required for this data, we chose omit this feature for the time being. The Venn diagram would have shown how much overlap the author's community has. Attempts were made to acquire this data but Schmid's server[7] had issues with the requests requiring too much system memory which caused crashes and directly accessing the data from Microsoft would often take over 10 minutes and cost about $2 USD per journal, which is just not realistic to implement when each author usually has more than 5 journals.

The possibilities to extend this application are basically endless and the author hopes to ensure further employment within the university to further extend this application once his ongoing fight with the abomination called Typo3 is done.

We present our application AcademicCV, which allows the user to easily compare two scientific authors' based several metrics. We also introduced a new metric, an author's community size, a attempt to mitigate factors such as the impact factor on an author's performance. The application is available for public use[2] and will likely be open source in the future. We almost achieved all of our initial goals, while the omitted ones can be implemented by further improving the data acquisition infrastructure.

[2] http://buzz.informatik.uni-ulm.de

List of Figures

List of Tables

Bibliography

[1] M. Bostock, V. Ogievetsky, and J. Heer, "D^3 data-driven documents," *IEEE transactions on visualization and computer graphics*, vol. 17, no. 12, pp. 2301–2309, 2011.

[2] L. Egghe, "An improvement of the h-index: The g-index." ISSI, 2006.

[3] A.-W. Harzing, "Reflections on the h-index," vol. 1, pp. 101–106, 2012.

[4] J. E. Hirsch, "An index to quantify an individual's scientific research output," *Proceedings of the National academy of Sciences*, vol. 102, no. 46, pp. 16 569–16 572, 2005.

[5] A. Post, A. Y. Li, J. B. Dai, A. Y. Maniya, S. Haider, S. Sobotka, and T. F. Choudhri, "c-index and subindices of the h-index: New variants of the h-index to account for variations in author contribution," vol. 10, 2018.

[6] R. Rousseau, "New developments related to the hirsch index," 2006.

[7] D. Schmid, "Combining interactive exploration and search for navigating academic citation data," 2018.

[8] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-J. P. Hsu, and K. Wang, "An overview of microsoft academic service (mas) and applications," in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW '15 Companion. New York, NY, USA: ACM, 2015, pp. 243–246. [Online]. Available: http://doi.acm.org/10.1145/2740908.2742839

[9] A. W. Wilhite and E. A. Fong, "Coercive citation in academic publishing," *Science*, vol. 335, no. 6068, pp. 542–543, 2012. [Online]. Available: https://science.sciencemag.org/content/335/6068/542

Declaration

I, Stefan Wintergerst, matriculation number 904892, hereby declare that I created this work on my own. Except where referenced and credited to the original author, I have not used the material of others in my work.

Ulm, ...

Stefan Wintergerst