University of St Andrews

Reproducible analysis of the CORD-19 Software Mentions dataset

School of Computer Science - University of St Andrews

Master of Science Information Technology with Management

Patrick Kornek – 190031409

Supervisor – Dr Alexander Konovalov

Submitted on 17th August 2021

Abstract

A lot of software is developed and used for research purposes. However, citing software is challenging for researchers which is part of academic work. This dissertation aims to analyse the CORD19 dataset which incorporates software mentions. In detail, the dataset is extracted by a machine learning algorithm and related to coronavirus publications. The mentions and complementary information are analysed to output valuable insights. Consequently, the dissertation is accompanied by a collection of six Jupyter Notebooks written in Python. The first three notebooks focus solely on existing information of the CORD19 dataset and create meaningful findings. These verdicts are related to software mentions. To obtain further insights, additional information is analysed by the three notebooks which target fetched data from an API service. The analysis of external data enriches the available findings with visualisations and provides new viewpoints on the CORD19 dataset. As part of the analysis, anomalies are determined within the dataset. For this, improvement proposals are suggested to enhance the quality of the algorithm which generated the CORD19 dataset. The study provides its notebook infrastructure in a reproducible pipeline to promote recreation to readers that are interested in the project.

Declaration

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated.

The main text of this project report is 14,287 words long, including project specification and plan.

censored

Patrick Kornek 17th August 2021

Table of contents

1. Introduction

1.1 Problem description

In an academic context, researchers utilize resources to conduct their studies which are published and contain information about employed software. The dataset named “CORD-19 Software Mentions” focus on recent research related to coronavirus publications [1]. It was extracted by a SciBERT-based machine learning model from full texts of scientific papers [2]. This algorithm was trained on the Softcite dataset and outputs software mentions and its metadata [3]. The metadata is in an uncleaned condition and potentially provides further insights. For this, the project will evaluate the quality of the dataset which may be reported to the authors supporting them to enhance the machine learning model to extract software mentions.

1.2 Aims and objectives

From a viewpoint of the author, this project targets several objectives. First, the researcher is expected to acquire programming skills related to Python and its ecosystem to conduct the project. Second, the dataset named “CORD-19 Software Mentions” need to be cleaned and a descriptive analysis is conducted to form a basis for further tasks. Third, the report is accompanied by a collection of Jupyter Notebooks analysing the dataset following the principle of reproducibility.

Furthermore, each notebook features its own analysis goal. For instance, it must be considered to find the most used software. To show the data from a different angle, the observer is supplied with visualisations that are built on top of corresponding data. The analysis is not limited to existent data which means that additional information is considered via external services. Potential follow-ups can be conducted with the GitHub API to create an insight into software licenses.

1.3 Report Outline

This report is divided into several chapters to discuss various key aspects. The first chapter provides a broad overview of the thesis and introduces the reader to the topic. In chapter two, a context survey is elaborated to expand the projects context with background information provided by academic literature. Subsequently, chapter three captures the requirement specification for the implemented algorithm within the Jupyter Notebooks. In chapter four, the taken development approach is elaborated and justified for its adoption. Afterwards, chapter five considers ethics and its relevancy for this thesis. In chapter six, the design structure of the implemented algorithm will be indicated. How the implementation was tested and conducted can be obtained in chapter seven. The fulfilment of work with respect to the initial objectives is evaluated in chapter eight. In the end, chapter nine is summarising the outcome of the dissertation and considers future work.

2. Context survey

2.1 Background

Nowadays, science has more data available than before and faces the challenge of analysing it. Especially, genomic data about molecules, proteins or genes are more than doubling every year [4]. In academia, software is partially used to automate or simplify scientific processes. The more software is used in academia, the more important it is to have clear guidelines for the use and citation of software. The scientific community has no consensus on how software should be cited. It should be noted that different views and practices are depending on the academic field. Simultaneously, credit attribution for referenced literature is essential for authors of referenced software. As a result of the above circumstances, some individuals and groups have already published proposals in the form of publications [5].

In general, publications that consist of software and literature require consideration of various aspects. First, the audience needs to be supported to understand the basic principle of the interweaving of scientific knowledge and software artefacts. For instance, the connection of code and corresponding literature posits such a challenge. Second, this thesis introduces its audience to software citation principles and how they are applied to changing circumstances. From a coding perspective, the observer is presented with useful tools and implementation details considering performance. Based on the theoretical part of the dissertation, research gaps are identified to derive guiding questions.

2.2 Definitions

2.2.1 Open Source

The term open source is widespread within the domain of Information Technology and Computer Science. Indeed, the wording “open-source” was coined in 1998 after the company Netscape published its code for their browser [6]. Nowadays, many perceive open source as a free access license to software that is not a software license itself [7]. It is a description of corresponding licenses granting permission to users. To qualify as open source product, all of the following conditions need to apply [8]:

- Free distribution means that users are not restricted to sell or distribute the software as part of aggregated software. Moreover, the software can be combined limitless with various sources.
- The source code of the utilized software must be included within the distribution. Therefore, obfuscated source code is not allowed because programmers need to be provided with readable code.
- The derived work of the initial codebase is allowed. Due to further modifications, the code must be distributed under the same license as it was accessed.
- To credit the author of the source code, the integrity of the code must be granted by the license distributing the software.
- The open-source code does not restrict or discriminate peers which means that it is accessible and usable for everybody.
- Open-source software applies specific rights and corresponding licenses which are valid for all peers.

The heart of open-source projects is its communities which predominantly work decentralised on the source code. The communities are internet-based and collaborate voluntarily to develop code based on their persuasion. For instance, a possible incentive is that they implement code on the demand of themselves or their organization [9].

2.2.2 Software

Software is a common term for programmes and the associated data. It determines what a software-controlled device does. The hardware executes software and thus puts it into action. In detail, software is the accumulation of information that must be added to the hardware to make a software-controlled device usable for a defined range of tasks. For the term software, there has not been a standardised definition which means that various description exists. The interpretation often differs in detail depending on the author and context. In general, the term software regularly refers to programmes, but not to data [10]. However, the source code, data or documentation can also be included, depending on the definition [11]. The current standard IEEE 24765:2017 replaced IEEE 24765:2010 and contains the following definitions for software [12]:

- Software is a program or set of programs used to operate a computer.
- Software are programs and the associated documentation.
- Software are programs and, if applicable, the associated documentation and other data necessary to operate a computer.

Another aspect linked to software are artefacts which play a crucial role during the development of software. An artefact represents a result of a work process. Examples of such outcomes are files with source code as the result of a software development process or a text document as the result of defining requirements for a system [13].

2.2.3 Software types

To cite software, it is essential to distinguish between various software types which can be conducted by distinct identifiers and relevant metadata such as date and name. For categorizing software, different approaches are available considering metadata. In general, software types can be determined by layer, function, or distribution mechanism. Further software categories are classifiable by the status of instantiation, permission and publication [5]:

- For the permission status, software can be distinguished by its license such as open or closed source. Indeed, licenses for software are diverse but for classification two categories are important to present differences for citation. In comparison to open source, closed source does not grant the reuse or view of source code.
- Furthermore, software has a publication status which is either published or unpublished. The type of publication is linked to the formal process of archiving a copy of software via a publisher which allocates an identifier and proper metadata to it. Consequently, the software is preserved for the long-term on archival repositories such as `www.figshare.com` or `www.zenodo.org`. Software that has been put in a random place on the web cannot be seen as published because relevant and resolvable identifiers are missing. Moreover, publishing source code to GitHub must

be seen as online and unpublished due to the circumstance that GitHub does not support preservation for longer terms.

- The instantiation status of software separates between instances such as versions and ideas called concepts. Concepts can become a version by instantiation. For theory purposes, concepts can exist for practical reasons as ideas without an instantiation. Nevertheless, concepts are only citable if there is a corresponding version containing metadata and a resolvable identifier. In general, concept software can be considered as a collection of versions.

Next to the clearly defined software types, there are hybrid types that do not fit accordingly to one type. For instance, the software can have closed and open-source components or published and unpublished source code. Therefore, it is crucial to identify code components accordingly with its metadata. Furthermore, the software can be classified as a mixed-type object such as computational workflow or Jupyter Notebook [14]. These called mixed-type objects are used for research and are known as Research Compendia or Research Objects [15].

2.3 Related Work

2.3.1 Traceability

For researchers, there are different possibilities to link source code to affiliated papers posing challenges to maintain and create the optimal circumstances for such links. The goal of linking scientific knowledge to software artefacts is to enhance its accessibility, understandability, readability, and usefulness. Therefore, the traceability between scientific papers and their corresponding source code poses a vast challenge to ensure knowledge transfer [16]. A possible approach to ensure traceability is to use the machine learning algorithm which automatically detects dependencies between links to artefacts in repositories with papers [17]. Moreover, the increasing popularity of these services undermines the relevancy of traceability between scientific knowledge and source code artefacts. The algorithms are confronted with the circumstance that code and papers evolve within different containers [16]. For this, the Open Science Initiative of the Empirical Software Engineering Journal asks researchers to publish materials openly because transparency is required for machine learning algorithms to work [18].

Consequently, the linkage of software and science closes the gap between different scientific communities by connecting them. It can be observed that many domains have scientific repositories containing references to academic papers. For this, a conducted study reported that code comments are linked to academic papers [19]. For the knowledge transfer between scientific papers and software artefacts, various types are identified but the traceability challenge persists. Especially, the identification of links, their creation and maintenance obstruct knowledge transfer and reproducibility. Indeed, there are no conventions on link localisation for proper tracing. For maintenance, links face decay issues if papers or software are updated because it is complex to announce modification on the correlative knowledge track. For traceability, the technological and conventional implementation of links between software artefacts and research papers are not manifested yet. There is a clear demand for tools for comparing and managing software artefacts and corresponding papers. Moreover, techniques displaying metadata of code and corresponding papers depict an additional request [16].

2.3.2 Software citation principles

In recent years, the FORCE11 Software Citation Group published a paper proposing software citation principles to encourage researchers for a unified citation principle covering various disciplines. Their motivation for the introduction of such suggestions is established on the circumstance that research is conducted increasingly digital. Consequently, knowledge is more available within digital components. Moreover, conventional knowledge carriers like books or papers are digitalized and their reproducibility is enhanced. Therefore, the following six principles are proposed [5], [20]:

1. Importance: Simultaneously, software is such as literature a component of research that requires citation. Independently, literature and software must be dealt the equal importance as bibliography record of research. Consequently, it should be prevented that software is omitted from scholarly records. Based on academic principles, software must be a product to cite such as conventional academic literature.
2. Credit and attribution: The principle of academic attribution and scholarly credit must be awarded to the programmers of software equally like to author of papers. For this, it needs to be considered that software attribution styles vary depending on the software.
3. Unique identification: When software is cited, it is requested to be in form of a machine-readable way that is unique and interoperable for the domain. Consequently, community members of their domain must be able to work with the citation. Moreover, a form that generally supports researchers is more beneficial.
4. Persistence: The cited software must be affiliated with metadata and unique identifiers which persist for the software's lifespan and beyond.
5. Accessibility: The cited software and its metadata must be accessible by its citations for researchers and machines for reference purposes.
6. Specificity: The software citation must facilitate access to the specified and utilized version of the software. Therefore, the identification of software and its corresponding version is requested to be as specific as possible by employing variants and version numbers.

The FORCE11 Software Citation Group specifies the identification of operating software during research and its citation within a paper as challenging [5]. For instance, Howison and Bullard specify seven distinct types of how software was mentioned within academic work [21]:

- *Citation of publication*
- *Citation of user's manual*
- *Citation of name or website*
- *Instrument-like*
- *URL in text*
- *In-text name only*
- *Not even name*

For several reasons, the inaccurate use of software citations posits a concern to academic researchers. First, research software is directly related to the academic workflow which requires referencing when it is utilized. Second, software developers must be rewarded for their contribution by crediting it the same way as academic work is honoured by citations. Third, citation of software is required to introduce peers to software and allow them further exploration of source code. Besides citation, the corresponding configuration settings for software is essential to reproduce outcomes [5].

In Information Technology and Computer Science, data is a term which can be processed by machines such as computers. Based on this description, software is a type of processable data that needs more consideration than citing data [10]. Compared to the credit system of papers, software does not need a distinct credit system because the credit itself is required and not a system. For instance, Katz and Smith suggest a transitive credit which provides an entire representation of citations and corresponding credit [22].

Furthermore, different approaches exist among various communities and domains considering software citations within research work. A reliable source for related work is the UK Software Sustainability Institute which accommodates a blog for its community endorsing discussion. For instance, Jackson reveals perils occurring when researchers are citing publications. Moreover, Jackson advocates citing software by guiding researchers [23]. It is notable that authors usually obey existing citation instructions which means that software providers should indicate reference guidelines. Otherwise, the user is less likely to cite employed software [24]. An alternative of not presenting citation principles leads to the circumstance that peers implemented community guidelines for software referencing. Similarly, specific domains encourage authors to publish their work within software papers which elaborate parts of research code and increase the capability of the artefact to be cited. Software papers are usually published in combination with the source code. The FORCE11 Software Citation Working Group remarks that suggestions for metadata standards affect software referencing. Nevertheless, this issue recorded interest among the community which means that there is demand for conventions specifying research software and its metadata [5].

In addition to the demand for proper citing procedures, there is controversy about the mentioned software citation principles. First, the principles show how software must be cited but do not clarify what software is eligible for referencing. The FORCE 11 Software Citation Working Group leaves this decision to authors and suggests a minimum of guidelines for identifying software based on citations. In the case of derived software, there is the possibility that modified code is cited but the reference to the initial codebase is missing. On one hand, the result of software citation principles is that peer reviewers are encouraged to reproduce outputs. On the other hand, software principles solely intend to identify utilized software artefacts and do not check if a cited paper was peer-reviewed. The implementation of software citation principles will impact the total number of references within journal articles. To a certain extent, journal guidelines limit the total number of citations which contradicts the principles of software citations. Indeed, the FORCE 11 Software Citation Working Group disagrees with this issue and recommends increasing awareness of software citations by diminishing limits and introduce advice for citing software [5].

Another aspect from the viewpoint of the FORCE11 Software Citation Working Group is the identification of software artefacts. The group demands a methodology which is "machine actionable, globally unique, interoperable, and recognized by a community" [5]. On top of this demand, Starr et al. request an identification principle that is feasible by a machine-learning algorithm on the web to enable persistency [25]. Moreover, `doi's` are suggested as the optimal identifier as they are widely utilized and accepted as a publication standard. In detail, a unique identifier corresponds to a software version which means that different identifiers can point to the same software but vary by version. Possibly, various identifiers pointing to the same software version leads to divided attribution and credit. For this, there are three relationships between software versions and their identifiers [5]:

- The latest version of software is referenced through an identifier.
- The software is identified generally including all versions.
- A particular version of a software part is identified.

Apart from the described cases, an identifier guiding users to a landing page that contains information about software and corresponding metadata is beneficial. A persistent landing page annotating software is crucial for longevity reasons. For citing software, dozens of cases exist which the researcher needs to obey and consider metadata accordingly [5].

2.3.3 Citation Management Software

In an academic context, citation management software is widespread and used by various stakeholders such as students, librarians, or lecturers. The functionality of this software is independent of the range of available offers. For most programmes, users can save references to a personal database, select a citation style, and create a corresponding bibliography rapidly. For instance, popular citation management programmes are RefWorks, Mendeley, Zotero or EndNote. These services differ mainly by monetary aspects and their peer groups. Indeed, institutions like universities and its library pay for licences which are used by affiliated students. Of course, there is the chance for peers to purchase licenses for themselves. In general, the user receives limited cloud storage where references are saved which requires internet access [26].

Furthermore, citation software varies by functionality featuring basic and advanced functionality. Some providers implemented solutions to extract metadata from PDFs while others did not. The basic principle is that users need to type manually collected data into input fields. More advanced solutions can automate imports of missing metadata and allow groups to collaborate by sharing references.

The workflow of citation software requests users to connect their user accounts to browser plugins. Consequently, the range of citation management software evolves by providing solutions for decentralised work environments such as groups and transforms itself into a cloud-based service [26].

2.4 Useful tools

2.4.1 GitHub

GitHub is a service that hosts code repositories using the git version control system and allows peers to collaborate on it [11]. The company providing this service is headquartered in San Francisco and was bought in 2018 by Microsoft [12].

The focus of GitHub is not on the project as a collection of source code, but on the user with its source code databases, the so-called repositories. The creation and merging of forks particularly facilitate the contribution to other projects. To contribute to a project, the repository is forked, and the changes are added as a request. Subsequently, the owner of the original repository decides whether the changes should be adopted via merging code or rejected [13]. Both the creation of publicly viewable and private repositories is feasible after free registration. One of the latest features of GitHub is the incorporation of citation files. GitHub endorses users to add citation information to repositories. Therefore, a `CITATION.cff` file must be added to the repository. This file informs researchers how the work should be cited [27]. Moreover, the `CITATION.cff` files consist of plain text which is human- and machine-readable [28]. For this thesis, a private repository on GitHub will be used to maintain a version-controlled structure of the implemented code.

2.4.2 Jupyter Notebook

In an academic context, published code and its outcome are required to be reproducible. For researchers, it is challenging to work with data, conduct statistical tests, depict visualisations, and administrate simulations. The more specific code is implemented by researchers, the less likely the code is published following reproducibility. Alternatively, researchers incorporate computational content in form of human language within their publications. Moreover, code is published as a supplement to existing reports which is hindering observers to see a relationship between code artefacts and text. Over longer periods, the academic work is at risk of becoming inconsistent. Therefore, Jupyter Notebook is a considerable solution for the stated challenges. It is a readable format for researchers to display explanations, code, and results. The structure of notebooks provides researchers with a framework to shift their scientific workflow to a reproducible format which accommodates interactive visualisations and details of the computation. For this, notebooks incorporate code within cells which can be run and modified individually. Subsequently, the result of each cell is displayed accordingly to the procedure read-evaluate-print-loop below each cell and saved within the document. In most cases, the output of a cell is plain text which limits functionality for researchers. Nevertheless, notebooks can display rich output like equations or graphics. For instance, Jupyter Notebook as an open-source project is compatible with various programming languages and can be accessed through web browsers. Due to its accessibility through web browsers, Jupyter Notebooks can be run locally or on a remote server. Jupyter Notebooks have the extension `.ipynb` and store their content as JSON files on devices. Due to the JSON structure, external software can easily access the file and conduct modifications. For sharing purposes, the Jupyter project provides a plugin which can convert notebooks to other formats such as PDF or HTML. This provides an observer with the comfort to access implemented code without installing and creating a Jupyter environment. Furthermore, web services exist to share notebooks environments for execution in combination with code on GitHub. Consequently, viewers can directly verify and interact with scientific code which is preservable for academic publishing [29].

2.5 String comparison

The comparison of strings is an important challenge for scientific work as it affects various academic domains. Due to applications containing key information expressed in sequences such as signal processing or text retrieval, the ability to quantify differences between strings posits a high value for researchers. For this, the most compelling methodology to compare strings is the Generalized Levenshtein Distance (GLD). The GLD focus on a varying amount of distance measures to state the relationship between strings. Indeed, numerous edit operations such as substation, deletion or insertion of characters or symbols create a distance. From another viewpoint, this measure can be called edit distance which describes the minimal cost for changing one substring to another. Therefore, the transformation of a string will be conducted by a series of edit operations with varying weights. On the one hand, the GLD is beneficial for the computation of pattern recognition or error correction. On the other hand, it is not recommended to apply GLD for noisy strings with a certain length. For this, it is crucial to understand that errors during a comparison of long strings are less decisive than in a comparison of short strings. A possible approach to circumvent this disadvantage can be conducted by normalization of GLD [30].

Alternatively, fuzzy string matching is another way to compare strings which have functional similarities to GLD. In general, fuzzy uses the edit distance of GLD to identify given patterns. Correspondingly to GLD, fuzzy investigates the count of primaeval operations to transform the string to a match. Typically, fuzzy is used for filtering spam and spelling error detection. In detail, fuzzy consists of a library named Fuzzy-Wuzzy which investigates string similarity among sentences or substrings and provides the outcome as a ratio between 0 to 1. Therefore, an outcome near to 1 means that the strings have a high resemblance. Whereas a result towards 0 shows less analogy to each other. The Fuzzy-Wuzzy package consists of four match logic standards [31]:

- Ratio: The ratio function is identical to the Levenshtein package which computes the distance ratio between two sequences such as strings.
- Partial Ratio: The partial ratio works with an optimal partial logic and investigates matches linked to the best substrings. By seeking the best matching substring, the partial ratio inspects the length of substrings and weights shorter strings more than longer ones. Moreover, the partial ratio does not obey the order of the substrings which potentially leads to poor results.
- Token Sort Ratio: The token sort ratio is beneficial as it prepares strings by removing punctuation and setting them to lower case. Before the comparison, the strings are sorted alphabetically. A remaining disadvantage is that strings of varying lengths are less likely to be matched.
- Token Set Ratio: In addition to the tokenization of strings such as the token sort ratio conducts, the token set ratio performs a comparison which checks for intersection and remainders. Compared to the token sort ratio, the token set ratio is more predestined for comparisons with different string lengths.

From an efficiency viewpoint, the Fuzzy-Wuzzy package slightly outperforms GLD. Due to four different matching logic variants, Fuzzy-Wuzzy provides more flexibility to its users. The Fuzzy-Wuzzy standards required consideration because they vary by performance [31]. Moreover, the package for Python provides a module named process. This module provides the functionality to investigate the highest resemblance “out of a vector of strings” [32].

2.6 Derived questions and research gaps

Publications linked to software citations have been published covering various aspects. Overall, researchers have investigated traceability, citations principles and citation management software. Nevertheless, it is important to conduct further studies by investigating software mentions within research papers.

Therefore, this thesis aims to conduct a reproducible analysis of the CORD-19 Software Mentions dataset.

The dataset is crucial for the related work because it has a wide variety of uncategorized software mentions which requires a dedicated inspection. Next to the analysis of software mentions, the dataset contains additional rows which will be investigated for insights. For an extended view of the dataset, further information will be requested from an external API service. Thus, this project aims to answer the following questions:

Q1: How clean and examinable is the quality of the CORD19 software mentions dataset? Does the dataset contain columns that cannot be analysed?

Q2: What is the composition of the software column within the CORD19 software mentions dataset? How is the distribution of software? Which is the dominating software mention?

Q3: What further insights can be created when extra information is fetched from external API services?

3. Requirements specification

3.1 Requirements linked to thesis

The following requirements should be obeyed during the creation of the source code accompanying this report:

- The source code should be reproducible which means that peers can recreate the outcome.
- The source code should be annotated to provide additional insights for peers viewing the codebase.
- The dataset should be cleaned up which makes it usable for further investigation.
- The source code should analyse the dataset descriptively.
- The analysis should provide visual insights.
- The analysis should output information about the dataset such as the most used software packages. It would be beneficial if the analysis categorizes software.
- The codebase should connect to external resources and incorporate additional metadata into the analysis.

3.2 Software Sustainability Institute

The Software Sustainability Institute has a considerable impact on this project which requires explanation and clarification. A team of peers with expert knowledge in research facilitation, software development, programme management and publicity engagement contribute from locations such as the Universities of Southampton, Oxford, Manchester, and Edinburgh. The institute follows the motto “Better software, better research” and promotes software advancement in various research domains to enhance the quality of research around the globe. From the viewpoint of the institute, sustainability is considered as an important aspect representing the unlimited availability of software. As a result, utilized software is updated and improved constantly for the future. Their knowledge facilitates the process of preserving research software with more than quality code. Indeed, the institute proposes practices to researchers for using and creating software. Moreover, they consult researchers for a cultural transformation within their communities. The Software Sustainability Institute assigns itself the following four fields of activity [33]:

- Promoting the growth of research communities for exchanging and spreading expert knowledge.
- Investigating the use of research software to provide insights to other researchers.
- Publishing guidelines for widespread use of software practices for research software.
- Volunteer for guiding and training of researcher communities by enhancing the acknowledgement of code in research.

The manifesto of the Software Sustainability Institute states that software portrays a major contribution to research which enhances the frontier of human knowledge. Moreover, they have the opinion that the full advantage of software in research will spread when software is used as research output. For this, the researcher should have access to software and training to utilize the benefits of code for their research. Consequently, the institute is convinced that practices for software lead to benefits such as reproducibility of research and software. They perceive themselves as representatives for developers of research software and understand the demand of software users. Especially, the acknowledgement of research software, the education of research communities and the peers implementing research software are crucial interest groups. Likewise writing a journal for research purposes is credited, the development of software should be honoured too as it requires a demanding skillset. Therefore, the scientific method of reproducible research is based on software-generated results and should experience more focus due to its reliability and reproducible procedure [34].

There are various ways to contribute to the activities of the Software Sustainability Institute. One of them is to participate in their Collaborations Workshop which is joined by different peers from distinct countries. During the Collaborations Workshop 2021, a hackathon was conducted. This hack day focused on the creation of insights linked to research software that was used for coronavirus related research. The project is called "Habeas Corpus" and is useful for the research community including the Software Sustainability Institute. According to the Habeas Corpus contributing guidelines, this thesis will participate in the following two points [35]:

- Investigate the issues published within the GitHub repository and contribute by solving them.
- Use the data to create an analysis and inform the Collaborators of the hack day about the outcome.

Both tasks fit the already defined requirements and are a component of this thesis. After the submission of the thesis, the implemented Python code will be submitted in form of a collection of Jupyter Notebooks as a pull request to the open-source project. The development of the software artefacts follows the guideline of the Habeas Corpus repository to match the demand accurately [35].

4. Ethics

This project considered the preliminary ethics self-assessment form and answered all questions with “No”. Therefore, the ethics self-assessment form is attached to Appendix A - Preliminary Ethics Self-Assessment Form. The investigated dataset contains the following metadata [1]:

- Identifier of paper
- Digital Object Identifier of the article
- Title of the article
- Provenance of the article
- Publication date of the article
- Journal short name
- URL(s) of an article
- Software mentions extracted from article full-text

Consequently, the dataset does not consist of the personal data of individuals and is not retrieving them for this project.

5. Design

The implemented Jupyter Notebooks can be used to analyse software mentions. Moreover, the algorithm is not restricted to software mentions which means that the code can be adapted to other terms or subjects. For this, the project is separated into three Jupyter Notebooks written in Python. Each of the files focuses on its own goal providing various insights for the observer.

For the further analysis, the dataset is connected to an external API which provides new data. Consequently, fetching a third-party service and its examination is conducted within three additional notebooks lifting the total number of notebooks to six. The files and their dependencies are described subsequently:

- `CORD-19-explore-dataset-cs5099`: This Jupyter Notebook file works independently and investigates all columns of the connected dataset to search for inconsistencies. First, this script should be used by the observer to connect to the dataset and obtain information about the quality of the data. As a prerequisite for the analysis, the script explores the existence of the global property value Not-A-Number (NaN). Moreover, all columns are searched for duplicates and checked for their significance. Indeed, this script can be seen as an additional insight towards the goal of software counting and classification which is conducted by other files.
- `CORD-19-software-counting-cs5099`: This Jupyter Notebook works independently and is predominantly responsible for proper counting of software mentions linked to the column software of the CORD19 dataset. For reproducibility purposes, the user can replace the software column with another to focus on varying subjects and terms. To count software mentions, the algorithm uses built-in pandas functionality. The Levenshtein extension is used for further string comparison downsizing the number of software mentions.
- `CORD-19-software-classification-CS5099`: This Jupyter Notebook is dependent on the execution of the notebook named `CORD-19-software-counting-CS5099` because it requires software mentions as an input. The goal of this script is the classification of software based on the defined CSV file named `software_categories_CS5099.csv`. This CSV file contains various categories for software and was created manually. From a viewpoint of reproducibility, the user can modify the CSV file by creating their categories and the algorithms conduct the classification.
- `CORD-19-collect-scopus-data-CS5099`: This Jupyter Notebook runs independently and is connected to the Elsevier elsapy API to fetch SCOPUS data. Therefore, the column `doi` is used to create a specific API request. As a result, the obtained data is adapted to the ethical standards of this dissertation and stored within a DataFrame. Due to the circumstance that the Elsevier database regularly updates and inserts new information an extended functionality is implemented to the notebook. This feature focus solely on entries that were fetched from the API and did not return any information. Indeed, this feature continuously fetches None values until they are filled with data.

- `CORD-19-analyse-coredata-CS5099`: This Jupyter Notebook is dependent on the outcome of the notebook named `CORD-19-collect-scopus-data-CS5099` because it analyses a part of the fetched SCOPUS data. Based on the obtained core data, this notebook targets to provide insights into the literature types of the publications. Moreover, each entry is affiliated with a citation count which presents the relevance of publications within the dataset.
- `CORD-19-analyse-affiliation-data-CS5099`: This Jupyter Notebook relies on the execution of the notebook named `CORD-19-collect-scopus-data-CS5099` because it examines affiliation information such as countries. This notebook aims to create a geographic insight into the dominating contributors to the dataset.

For the optimal user experience, it is recommended to run the script `CORD-19-explore-dataset-CS5099` to obtain a broad overview of the existent columns. Next, the software mentions need to be counted and unified. After the count, the classification script can be conducted to spot the distribution of software mentions. Simultaneously, it is required to run the script `CORD-19-collect-Scopus-data-CS5099` before analysing the fetched information.

6. Implementation

6.1 Methodology for Jupyter Notebooks

The implementation of the Jupyter Notebooks follows a varying methodology depending on the use case. In general, all notebooks obey guidelines for sharing and writing computational analyses. Especially, when projects evolve and scale in complexity, it is challenging to create descriptive information and entry points for researchers to understand the subject and reproduce the outcome. To accomplish a standard addressing the described challenges, this project considers best practices. Therefore, reproducibility is the key for academic work and posits a minimum principle for computational notebooks. Therefore, the rules for implementing Jupyter Notebooks can be seen as an interactive workflow. The cycle is elaborated based on its phases. It starts with documentation and organizing aspects. Then, quality standards are implemented within the notebooks. When the implementation is finished, the generated knowledge is published for further exploration and animates external researchers for participation. In detail, the obeyed guidelines are elaborated subsequently [36]:

- The interaction between researchers and their audience is supplied by a Jupyter Notebook. Next to the code, the researcher can introduce the topic with text components and explain the results. For the audience, the cells must be connected and illustrated to promote comprehension. Therefore, proper documentation is key to provide various levels of clarification addressing each audience. Next to the observer of this thesis, the dissertation creator is the primary audience for this thesis. In the future, the author needs to reproduce and understand the outcome of this work. Moreover, this project and corresponding knowledge is a basic layer for prospect work.
- During the academic process of writing a thesis, the documentation of results is not the only aspect to consider. Due to unforeseen events such as dead ends, proper notes stating all explorations is essential. A complete sheet containing notes eases comment writing and supports the researcher to memorize working steps. Moreover, notes are usable for further actions such as creating a presentation of a coding project. In the end, not used or unnecessary notes can be discarded. From a viewpoint of an author, it should be avoided to add explanations after finishing the code and analysis. The risk of missing important information such as modified parameter values or forgetting the citation of copied code lowers the quality of the outcome. Therefore, a proper workflow accompanying experiments are a solution which means that notebooks should be annotated and organized. For instance, a compelling way to present figures to readers is to use plots to gather meaningful insights.
- The interactive environment of Jupyter Notebooks can be run with notebooks consisting of one or several cells. Consequently, notebook cells can be fragmented or overloaded with code convoluting comprehension. Likewise writing text paragraphs, code is modularizable within a cell. For this, cell divisions are a possibility to distinguish the code of two different tasks generating clarity. Moreover, each

notebook cell should perform one action which is accompanied by a markdown description clarifying the step of the analysis and foster comprehension. Alternatively, the notebook can be structured with markdown headers which divide topics into sections and provides orientation for the reader. A series of short notebooks can be created out of a long notebook to form a navigable experience.

- For reusability, the modularization of code is crucial to assign functionality to a dedicated location within a notebook. This means that there is no space for duplicate code. Moreover, reusable code can be implemented within a function to be called from any cell. When a function is valuable for other projects, it might be helpful to transform the code into a package or module. Indeed, modularization is beneficial as it saves storage and facilitates maintenance.
- Next to code, the recording of dependencies of a notebook is essential as running code in the future requires proper referencing to utilized libraries or packages. Therefore, package managers such as `Conda` or `pip` support the process of downloading specific components and notice used versions which are stored within the manager's environment. Furthermore, it is suggested to implement code within environments that grant the correct recording of dependencies. The risk of undocumented dependencies can be avoided by using notebook extensions.
- Due to the interactive nature of notebooks, code may be accidentally removed, or the code comprises bugs requiring further considerations. To address inconsistencies within the code, it is wise to use version control systems to govern the history and investigate when a bug was incorporated into the code. Therefore, the widespread version control system Git distinguishes Jupyter Notebooks based on their differences in JSON files which contains metadata and corresponding code. This means that the differences are not displayed on a graphical user interface of a browser. A possibility to show changes in a user-friendly way is to apply diffing tools.
- For additional work, it is beneficial to implement the Jupyter Notebook as a generalized pipeline. Based on different datasets, the parameters and input data are flexible which can generate a varying outcome. For a new analysis and as a prerequisite, variables need to be visible at the top of the notebook facilitating changes. Furthermore, regularly restarting the whole notebook is a must to recognize cell removal of fundamental content.
- For a reproducible project, it requires sharing the utilized dataset along with implemented code. Otherwise, readers of the code cannot reproduce results and conduct an interpretation. In some cases, the dataset is sensitive or cannot be shared due to its large size. Therefore, complex datasets should be broken down into smaller parts without constraining interpretability or reproducibility.
- When researchers investigate external notebooks, it is beneficial to provide a design that allows convenient reading, running and exploration. For this, the notebook can be published on public repositories containing a README file and an open-source license encouraging the reuse of the code.

6.2 Data preparation

The analysis of a dataset requires pre-processing such as data parsing of raw data. Due to unusual structures or formats, the dataset cannot be analysed from the beginning and needs to be transformed to an analysis friendly form. Especially, the sources of a dataset are potentially corrupted, inconsistent, or incomplete. For this, computational programmes, or tools remedies the source and enable a solid base for analysis. The parsing of data can be facilitated with the following approaches [37]:

- To parse row data, it is beneficial to work with assertations addressing the data format. The assertations are created to violate raw data checking format inconsistencies. The more assertations are created, the more modifications by code can be conducted to parse data. Due to assertations, errors will be reported by the computing environment which indicates that there is further room for improvement. By maintaining a stringent assertation policy, the data format will have a satisfactory condition for analysis.
- When the record contains missing or broken records, it is recommended to display the corresponding information. Instead of ignoring it, the researcher can print the affected raw data accompanied with a warning allowing inspection. Another approach is to count the number of affected records and skip them. Nevertheless, skipping records should be the last practice because the analysis quality can be negatively impaired. Therefore, the researcher should consider how many records would be skipped compared to the number of all records. This means that skipping 1% of the records is more justifiable than 20%.
- The presence of categorical variables promotes the approach of counting its occurrence. Therefore, pandas provides built-in functions. Alternatively, the counter can be manually implemented which requires consideration of new occurring or invalid values. For instance, new values are printed out to notify the researcher of their existence and invalid values require a dedicated inspection to prevent misinterpretation.
- Due to the circumstance that large raw data requires more computation time than small, the risk of crashes increase. Therefore, enabling the start of data parsing at all positions is beneficial because the algorithm can be restarted flexibly saving time. As a prerequisite, the algorithm must print out information about the current computed record which can be used for a restart or an investigation when errors occur.
- Another time-saving approach is to develop a data parsing method that investigates solely a subset of the original data. Especially, at the beginning of the project, working with a random selection of raw data boosts the progress. Subsequently, the selection can be enlarged when the algorithm passes early stages and meets a robustness goal.
- When the researcher is provided with a large amount of storage, the redirect of log files can be considered for inspection purposes. Moreover, masses of space facilitate the storage of cleaned and raw data. As a result, each record of the cleaned data set can be compared to the raw dataset which allows inconsistency inspection.
- After each parsing step, the cleaning process and the integrity of the data must be verified. Therefore, researchers are required to write their own algorithms or use available functions to ensure a high quality of the data which is used for analysis. It is

important that the data has the proper format before starting the analysis. In contrast, a falsely conducted cleaning process can potentially output misleading results.

6.3 Exploration of the dataset - CORD-19-explore-dataset-CS5099

First, a general check of the dataset is conducted by investigating the number of NaNs in all columns. Therefore, built-in pandas functions are utilized to obtain the total amount of NaNs. Additionally, a self-designed function inspects each column and returns the number of NaNs per column. The exploration of NaNs within the DataFrame is crucial to check the expressiveness of specific columns. If a column contains a high portion of NaNs, the columns usefulness for analysis requires consideration. The general inspection continues by investigating the average column length inferring descriptiveness. This process is supported by the modification of built-in pandas functions. The thesis concludes that a longer column length is more descriptive than shorter. Especially, the analysis of string originated columns such as `journal` or `title` rely on length. Both, column length and number of NaNs are a considerable prerequisite for the analysis. For the dedicated exploration, a pandas series is created and assigned to each column. As a result, the notebook contains eight series named:

- `paper_id`
- `doi`
- `title`
- `source_x`
- `license`
- `publish_time`
- `journal`
- `url`

The series object has axis labels and saves its content within a one-dimensional NumPy array. Moreover, the labels are not required to be unique, and the object can be indexed by label or integer position [38]. This object supports the analysis because potential duplicates from CORD-19 columns could be stored in it.

Subsequently, the notebook inspects the integrity of each column and delivers additional insights. For the columns `paper_id` and `doi`, it is compulsive to see if all identifiers are assigned uniquely. Simultaneously, the column `title` is checked with a built-in pandas function for shared values and compared to specific substrings to find the density of COVID related publications. The columns `source_x`, `license`, `journal` and `url` are using own and built-in pandas functions to count the numbers. For the columns `url` and `journal`, it is interesting to find dominating publishers. Lastly, the column `publish_time` is accumulated to a DataFrame to obtain information about publication periods. In general, the presented series objects are researched for exploratory purposes concluding the algorithm quality which extracted the dataset. Due to differentiation, the series `software` is explored within the notebook counting software mentions.

6.4 Count of software mentions - CORD-19-software-counting-CS5099

The purpose of the notebook counting software mentions is to search for overlaps between various mentions and assign them correspondingly. To start with, the column `software` requires cleaning to contain one software mention per row. For this, an own coding algorithm is implemented to divide multiple mentions into one series. Moreover, the software mentions are cleaned to remove unwanted characters such as brackets or spaces. The cleaning of the software strings is crucial for comparison purposes granting a unified basis for the subsequent procedure. Again, the series object is beneficial as it provides the function `value_counts()` which returns a series containing counts of unique values [39].

In general, the process of counting software mentions works like a filter whereas all mentions start at the top and are minimized through various mechanisms to a smaller selection at the bottom of the filter. Before the comparison is initiated with the Fuzzy-Wuzzy algorithm, the strings within the series are capitalized to save a step for the fuzzy method. Due to performance reasons, the comparison of the substrings is conducted with a capped limit containing the current selection of the most counted software mentions. Depending on the dataset, the notebook provides all four fuzzy methods which can be selected for filtering. In the case of the CORD-19 dataset, the token set ratio is chosen due to its superiority over the remaining methods. Thus, the methods are discussed in terms of efficiency for the CORD-19 dataset:

- Ratio: The standard distance ratio is not efficient because the difference in string length is not properly considered. This means that a short and a long string with high similarity are less likely to be counted as similar.
- Partial Ratio: The partial ratio is less efficient as the standard distance ratio matches substrings based on their existence in another string. For instance, the mention `R` would match to `MICROSOFT` because of the existence of `R` in `MICROSOFT`.
- Token Sort Ratio: The token sort ratio does not work efficiently with the CORD19 dataset because it weights differences in string length heavily and skips similar substrings.
- Token Set Ratio: In terms of efficiency, the token set ratio obeys similarities of substrings independently from differences in string length. Moreover, the comparison of remainders and intersection boosts the performance which indicates this method as the proper choice.

All methods conduct a degree of mismatch which is limited to the maximum by setting the threshold of the fuzzy method. For the current dataset, a threshold of 84 is optimum as it works without matching the substrings negligently nor strictly. Moreover, a blacklist is created which collects all substring matches. Subsequently, the blacklist is deducted from the DataFrame containing all matches which are used to calculate position changes of the software mentions by its index position. As a result, the DataFrame contains three columns that give information about the software name, the accumulated number of matches and its position change compared to the situation before using the fuzzy method. For flexibility, the DataFrame is saved to an external file which can be run by other notebooks. This approach of storing the outcome to another file is beneficial because other notebooks can run independently from the initial notebook using the DataFrame for categorisation of software mentions.

6.5 Categorisation of software - CORD-19-software-classification-CS5099

The classification of software mentions relies on the PKL-file `software_mentions_CS5099.pkl` which was generated by the software counting notebook. This file contains accumulated information about software mentions and is updated automatically after each run of its notebook. Subsequently, the information is converted to a Dataframe and compared to a manually generated CSV file which provides software categories. The CSV considers the following categories:

- Statistics: This category is assigned when mentions are directly linked to statistical software which can be used for various fields.
- Bioinformatics: This category is allocated solely to software that has direct use-cases for bio-related subjects.
- Communication: This category is matched to mentions which are used for collaboration.
- Bibliography Service: This category is assigned to mentions which provide literature or are predominantly used for the storage of information.
- Operating Systems: This category is allocated to mentions which are a basis to run the software.
- Programming languages: This category is matched to mentions which are programming languages or affiliated packages such as frameworks.
- Uncertain: This category is created to match mentions which do not fit to one of the described fields.

All categories were considered manually and contain a different number of mentions which can be obtained from the file named `software_categories_CS5099.csv`. The choice for the CSV format is beneficial as it provides the user with the possibility to adapt and modify categories. Moreover, the target of this notebook is not to supply optimized and detailed categories which are narrowed down to the maximum extent. This notebook aims to be employed with a small number of general categories to grant a level of comparison between the choices. Indeed, the notebook facilitates the process of classifying software mentions and encourages its readers to create their categories. Consequently, this notebook works independently from the number of categories and their designation. For this thesis, the given categories are chosen to be valuable for the Habeas Corpus group.

6.6 Fetching new information - CORD-19-collect-scopus-data-CS5099

Fetching of new information relies predominantly on the Elsevier Developer Portal which provides a SCOPUS API service [40]. This service was implemented in the dissertation because it requires a `doi` as a unique identifier to return corresponding paper information. Therefore, the implementation of code was conducted by the modification of specific snippets of the elsapy documentation on GitHub [41]. Moreover, this notebook is dependent on the PKL-file `extra_info_CS5099.pkl` which stores fetched data. This file is updated automatically after information was requested and retrieved from the SCOPUS API.

For the fetching process, several considerations are crucial for this project. First, the API must be fetched with a reproducible procedure that consists of a series object holding `doi’s`. This object must contain each `doi` once and provide consistent order which is granted by the functionality `value_counts()` and `sort_index(ascending)`. Otherwise, neglecting sorting leads to a different order of `doi’s` within the series object. The order of `doi’s` in the series object is key as it stores the information in the same order as fetched. Inconsistences of fetching results in the circumstance that a `doi` is fetched and stored more than once. Moreover, fetching requires a valid API Key which needs to be regularly updated to circumvent API quota limits.

In terms of ethical guidelines, this project does not store critical data which may lead to the identification of persons. Indeed, the SCOPUS API provides sensible information which is counteracted with a control algorithm that solely stores cleaned entries according to the ethical code of this thesis. Before storing fetched data, the following information is removed from the API response [42]:

- `affilname`: This json-key contains information about the institution which published the corresponding journal.
- `affiliation-city`: This json-key holds the city of the institution which published the academic work.
- `dc:creator`: This json-key holds various subkeys which provide detailed information about the author of the publication.

Another crucial feature of this notebook is its flexibility towards unsuccessful API fetches. The SCOPUS API does not return information for all `doi’s` causing None values. There are various reasons for this circumstance which are linked to the responsibility of the author of this thesis and due to the SCOPUS API provided by Elsevier.

First, the author is responsible to call the SCOPUS API with a valid API key. Otherwise, the server does not respond with the requested information. Although valid information is available on the server, the API answers with an error message linked to accessibility. Therefore, reading the document fails and will be printed from the notebook to the user for notice purposes. Second, the SCOPUS API is a centralized service that can update existing information for papers. Theoretically, an unsuccessful API fetch at the beginning of the fetching period could have been modified after all `doi’s` were requested. Consequently, both events can occur due to the described circumstances demanding consideration.

The implemented algorithm within the `enrich_data()` function counteracts this incident by fetching solely entries that previously returned None values. In terms of being a sustainable researcher, this procedure is beneficial as it is collecting solely missing entries. Indeed, both fetching algorithms can be manually modified to fetch specific index ranges of the `doi` series object. Moreover, the SCOPUS API service is used efficiently saving resources by minimising the demand to the optimal extent. Consequently, the service is less employed than fetching all `doi’s` again which leaves more server resources for other users. Due to this algorithm, it is recommended to regularly run this notebook collecting newly available publication data.

6.7 Analysis of affiliation data - CORD-19-analyse-affiliation-data-CS5099

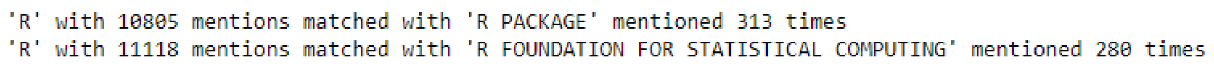
The analysis of the affiliation data is dependent on the PKL-file named `extra_info_CS5099.pkl` which is fetched by another notebook. This file contains accumulated information from the SCOPUS API which is divided into two sections. First, the affiliation data is investigated to show which countries are dominating publishers. Therefore, this notebook reads the fetched affiliation data from the PKL-file and stores it within a DataFrame for further processing. For expressiveness purposes, the number of None values must be considered by computing the ratio of None entries compared to the length of the DataFrame. Subsequently, None entries are dropped from the DataFrame. Due to the circumstance that an entry can contain one or more countries, helper functions were implemented which receive a dictionary with country names and return a DataFrame that holds solely one country per row. The countries are counted and transformed into a series which is sorted by the countries with the most publications. Moreover, the file is transformed to a CSV file which works in combination with Folium and the file `countries.geojson` [43]. As a result, a choropleth map is plotted to view countries and their affiliations. The implementation of Folium is conducted accordingly to its documentation [44].

6.8 Analysis of core data - CORD-19-analyse-coredata-CS5099

Simultaneously, the analysis of core data follows the same approach as the investigation of affiliation data. The fetched core data is read from the PKL-file named `extra_info_CS5099.pkl` passing the same procedure. None entries are inspected by their ratio to the rest of the DataFrame. Before implementing visualisations, None values are removed from the analysis. For plotting purposes, two types of charts are considered. First, a pie chart is incorporated into the analysis to display the different literature types. For the analysis, it is crucial to investigate which types are dominating the CORD19 dataset. Second, a scatterplot is applied to present the citation count of CORD19 publications. Therefore, the analysis inspects the proportion of publications with many citations towards academic work with less relevance.

6.9 Testing

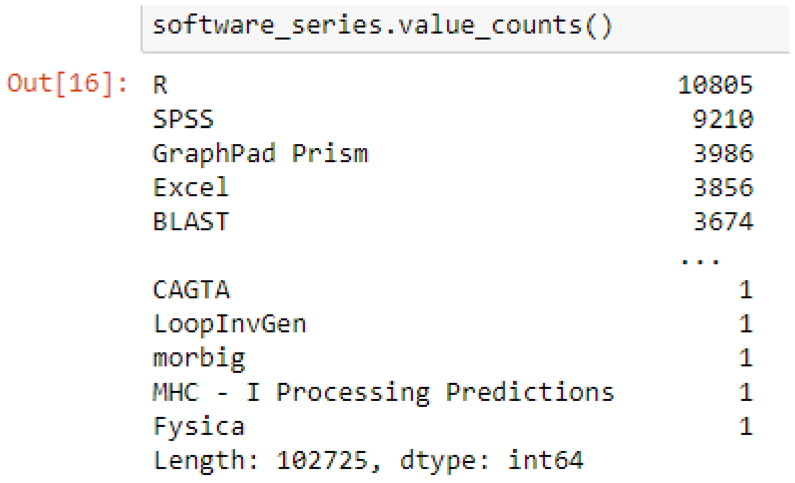
The implementation was tested with various metrics which ensure an appropriate quality standard. First, the code was verified with programming related techniques and functions. Second, the Jupyter Notebooks were checked for logical principles and purpose. Programming related techniques consist of conventional functions to control algorithms working properly. For instance, the `print()` statement was used to investigate the matching quality of software mentions. Especially, displaying string comparison matches to the screen is beneficial due to its informative output which facilitates further code improvement. Thus, Figure 1 presents an extract displaying matched software mentions.



```
'R' with 10805 mentions matched with 'R PACKAGE' mentioned 313 times
'R' with 11118 mentions matched with 'R FOUNDATION FOR STATISTICAL COMPUTING' mentioned 280 times
```

Figure 1: Logging string matches of software mentions

Further built-in functions such as `value_counts()` support the process of testing as it supplies direct feedback for unique values within a series or DataFrame and indicates the minimisation of software mentions. Figure 2 demonstrates how this process is supported with the `value_counts()` function.



```
        software_series.value_counts()

Out[16]: R                                  10805
         SPSS                                9210
         GraphPad Prism                      3986
         Excel                               3856
         BLAST                               3674
                                            ...
         CAGTA                                  1
         LoopInvGen                             1
         morbig                                 1
         MHC - I Processing Predictions         1
         Fysica                                 1
         Length: 102725, dtype: int64
```

Figure 2: Using value_counts() for counting software mentions

Logical principles were implemented according to an algorithm and its purpose. Therefore, each notebook has a goal which is formulated at the top of each file. For instance, the notebook counting software mentions is logically reviewed. This means that notebook cells that are responsible for summarizing mentions create objects with a decreasing number of distinct software mentions compared to previous cells.

Furthermore, the implemented functions within the notebooks usually contain parameters that were invoked with valid and invalid content. This procedure strengthened the robustness of functions and enhanced the performance. Another approach to verify the correctness of implemented code is to combine modified data with plots and observe the outcome. For this, it is crucial to inspect plots for specific ranges and detect outliners that might be a consequence of a bug.

For the notebook, which is responsible for fetching the SCOPUS API, several test procedures were conducted to grant a high quality for returned data. In a browser, the following URL was requested with valid API keys and `doi’s` to obtain corresponding JSON responses.

*https://api.elsevier.com/content/search/scopus?query=DOI(**<DOI>**)&apiKey=**<APIKEY>***

For a successful request, it is crucial to specify a correct `doi` and a functional API-Key which are marked in bold. This procedure facilitates the verification process of fetched data. Consequently, already collected data can be investigated. The previously received None entries can be manually accessed to verify their state. Due to this test practice, the different return cases of the SCOPUS API were considered. Usually, the API can return a full entry consisting of affiliation and core data. Next to obtaining no information for a request, the API can send back incomplete entries which include one of the categories.

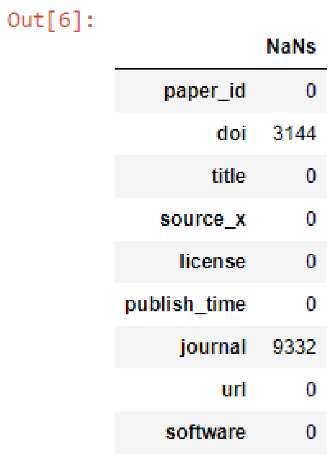
7. Analysis of the CORD19 dataset

Each Jupyter Notebook has a specific use case which is crucial for the overall outcome of this project. Due to their utility, the notebooks vary in cell length and complexity which creates different verdicts. Consequently, the notebooks are implemented to provide insights about the CORD19 dataset which are elaborated in the next subchapters.

7.1 Exploration of the dataset

7.1.1 General consideration of CORD19 columns

The exploratory notebook investigates generally the dataset and checks the quality of the algorithm which outputted the CORD19-file. Therefore, the columns of the dataset are inspected for duplicates and NaN's. Figure 3 presents the DataFrame containing all columns and its count for NaN's.



Out[6]:

	NaNs
paper_id	0
doi	3144
title	0
source_x	0
license	0
publish_time	0
journal	9332
url	0
software	0

Figure 3: CORD19 columns and its NaN's count

First, two columns contain a considerable amount of NaN's which need to be kept in view when the corresponding analysis is conducted. Moreover, general investigations continue by obtaining the mean length of each column to announce its benefit for the analysis. Especially, the mean length of specific columns is required to be sufficient to be counted as descriptive. The mean column length can be seen in Figure 4.

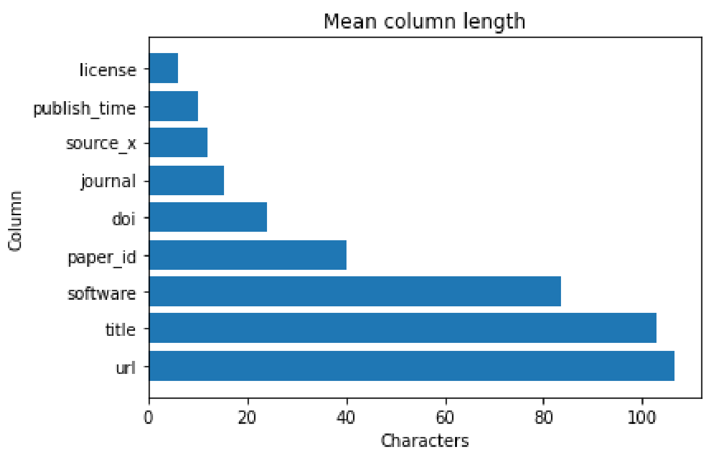
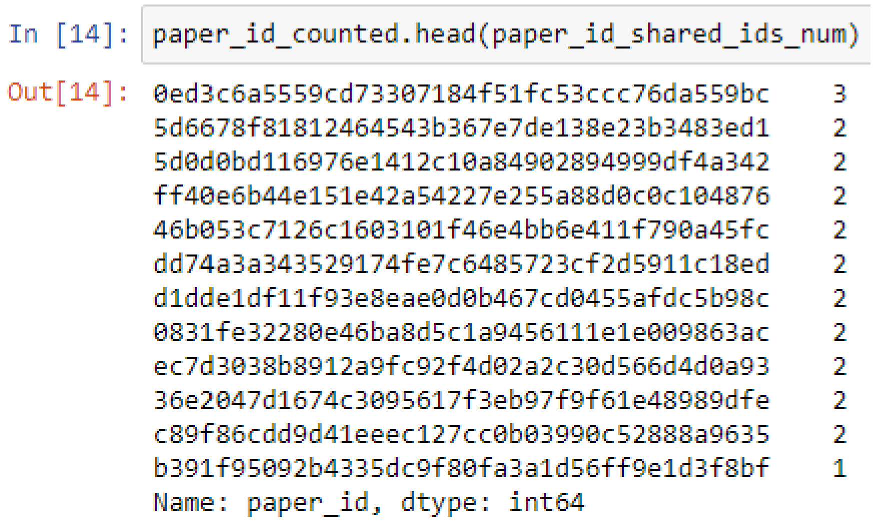


Figure 4: Mean CORD19 column length

As shown in the figure above, the column `title` and `url` have each a length beyond 100 characters. Due to technical prerequisites such as subdomains or paths, the column `url` is comparable long to other columns considering a further inspection. For instance, the `url` can provide information about the dominating publishers. Simultaneously, the length of the column `title` can be seen as descriptive and used to verify affiliation to COVID-related subjects by string comparisons.

7.1.2 Inspection of the columns linked to unique identifiers

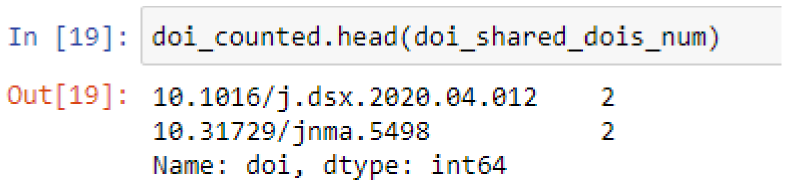
The quality of the algorithm which extracted this dataset is analysed with the uniqueness of columns holding identifiers. As a prerequisite of this analysis, it is assumed that the columns which contain identifiers must contain each value solely once. Otherwise, duplicate identifiers are an indicator of poor quality. Consequently, duplicates are likely to compromise the integrity of the analysis. The analysis of the column `paper_id` found out that this column contains a dozen duplicates. One of the duplicates appears three times and the remaining values are shared twice which can be seen in Figure 5.



```
In [14]: paper_id_counted.head(paper_id_shared_ids_num)

Out[14]: 0ed3c6a5559cd73307184f51fc53ccc76da559bc    3
         5d6678f81812464543b367e7de138e23b3483ed1    2
         5d0d0bd116976e1412c10a84902894999df4a342    2
         ff40e6b44e151e42a54227e255a88d0c0c104876    2
         46b053c7126c1603101f46e4bb6e411f790a45fc    2
         dd74a3a343529174fe7c6485723cf2d5911c18ed    2
         d1dde1df11f93e8eae0d0b467cd0455afdc5b98c    2
         0831fe32280e46ba8d5c1a9456111e1e009863ac    2
         ec7d3038b8912a9fc92f4d02a2c30d566d4d0a93    2
         36e2047d1674c3095617f3eb97f9f61e48989dfe    2
         c89f86cdd9d41eeec127cc0b03990c52888a9635    2
         b391f95092b4335dc9f80fa3a1d56ff9e1d3f8bf    1
         Name: paper_id, dtype: int64
```

Figure 5: Duplicated of the column `paper_id`

In terms of the uniqueness of identifiers, the column `doi` features better performance. The column `doi` accommodates two identifiers that share values and are displayed within Figure 6.



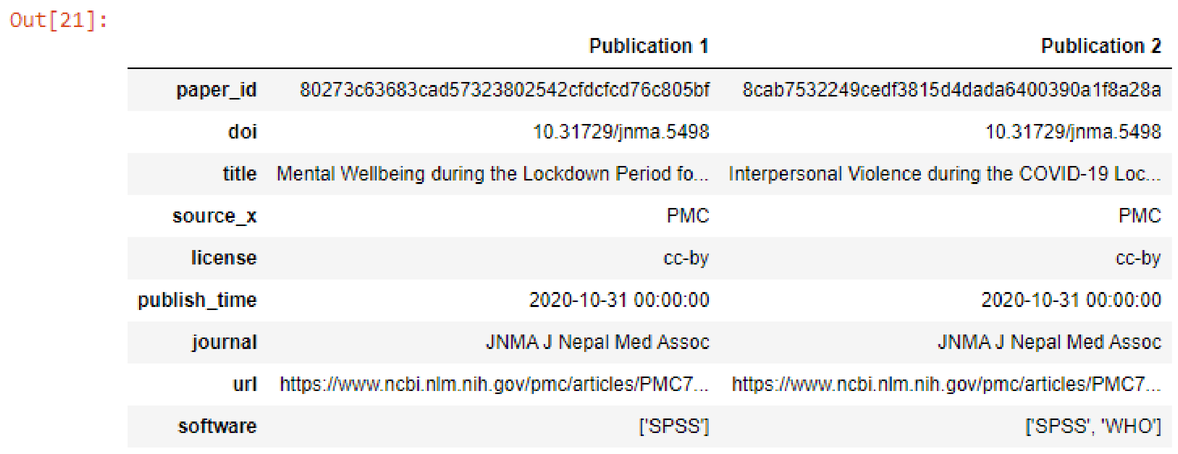
```
In [19]: doi_counted.head(doi_shared_dois_num)

Out[19]: 10.1016/j.dsx.2020.04.012    2
         10.31729/jnma.5498           2
         Name: doi, dtype: int64
```

Figure 6: Duplicates of the column `doi`

In comparison to the number of all rows of the dataset, these duplicates posit a sparse part which means that their impact on further analysis can be neglected. Consequently, both columns are verified for their usability and can be implemented for analysis purposes to the project. Nevertheless, this finding allows creating an improvement suggestion for the team which is responsible for the algorithm that extracted the CORD19-dataset.

- To enhance the quality of columns that store unique identifiers, an algorithm is required to check for shared values and remove duplicates accordingly.

Another finding was gathered for the duplicate publication with a shared `doi` named `10.31729/jnma.5498`. For this publication, all columns are identical besides `title`, `paper_id` and `software`. This leads to the assumption that both publications describe the same material, but the algorithm extracted varying software mentions. Both records are stored as publication in a DataFrame which are presented in Figure 7.


Out[21]:

	Publication 1	Publication 2
paper_id	80273c63683cad57323802542cfdcfcd76c805bf	8cab7532249cedf3815d4dada6400390a1f8a28a
doi	10.31729/jnma.5498	10.31729/jnma.5498
title	Mental Wellbeing during the Lockdown Period fo...	Interpersonal Violence during the COVID-19 Loc...
source_x	PMC	PMC
license	cc-by	cc-by
publish_time	2020-10-31 00:00:00	2020-10-31 00:00:00
journal	JNMA J Nepal Med Assoc	JNMA J Nepal Med Assoc
url	https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7...	https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7...
software	['SPSS']	['SPSS', 'WHO']

Figure 7: Unidentical duplicate in CORD19

This assumption inevitably leads to a second improvement suggestion.

- To enhance the quality of the column software, an algorithm must ensure reproducible outcomes. For identical publications, it must be guaranteed that identical software mentions are extracted.

7.1.3 Inspection of the column `title`

Subsequently, the column `title` is analysed to produce insights into the quality of the dataset. Simultaneously, this attribute is checked for shared values which leads to an ambiguous outcome. The identified shared values cannot be classified as duplicates because they tend to be a placeholder or show the poor naming quality of the papers. Subsequently, Figure 8 displays an extract of this circumstance.

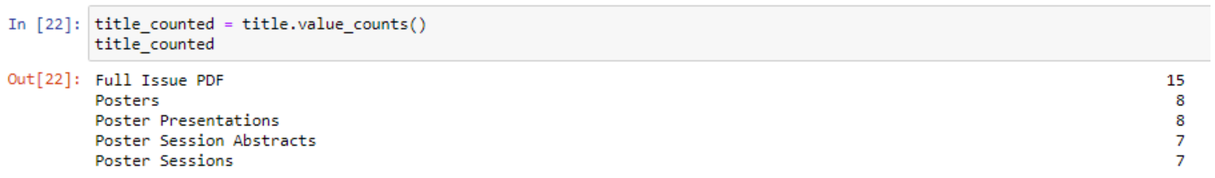

```
In [22]: title_counted = title.value_counts()
         title_counted

Out[22]: Full Issue PDF                                                                                    15
         Posters                                                                                            8
         Poster Presentations                                                                               8
         Poster Session Abstracts                                                                           7
         Poster Sessions                                                                                    7
```

Figure 8: Title duplicates in CORD19

For the analysis, the uniqueness of the title cannot be assured because there are papers that have placeholders or are assigned to default titles. Nevertheless, this posits a small percentage of all titles which encourages further analysis of this column. Next, the titles are compared to strings to inspect the density of COVID related publications within the dataset. Therefore, common words were elaborated and implemented in the comparison processing. Thus, the terms are listed:

- Covid
- Corona
- Lockdown
- SARS

As a result, 40% of the CORD19 dataset matches to one of these terms. Duplicate count between two or more terms is programmatically prevented. Nevertheless, it cannot be excluded that the missing 60% is somehow linked to COVID related publications. To display the relevance of potential COVID effects on society the following terms are compared to the title column:

- Economic
- Social

The intention of this comparison is not to obtain the absolute number of mentions. Indeed, this comparison targets the ratio between both terms to conclude its importance for the researchers and their audience. Therefore, a social related paper is 3.3 times more often part of the data than an economic-related subject. However, both terms do not have to be about COVID publications and could potentially investigate distant fields.

From an analytical viewpoint, the analysis inspected a dataset that is designated to incorporate COVID related publications but could not be proven to be mainly driven by this field. Indeed, the given procedure expected more output linked to COVID.

7.1.4 Inspection of the column `source_x`

The analysis of the column `source_x` leads to information about publishers. Therefore, it needs to be considered that several sources can contribute to one publication creating an overlap. The accumulated count of sources is shown in Figure 9.

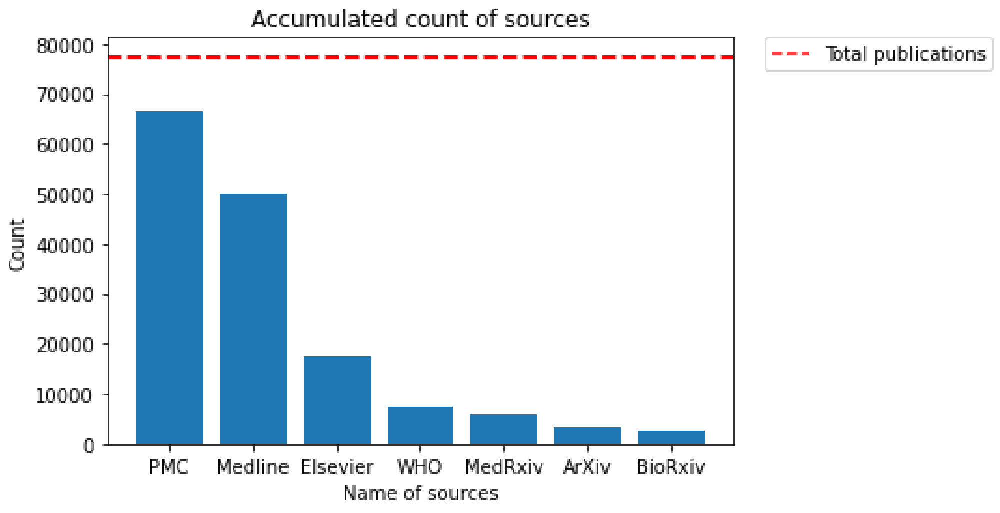
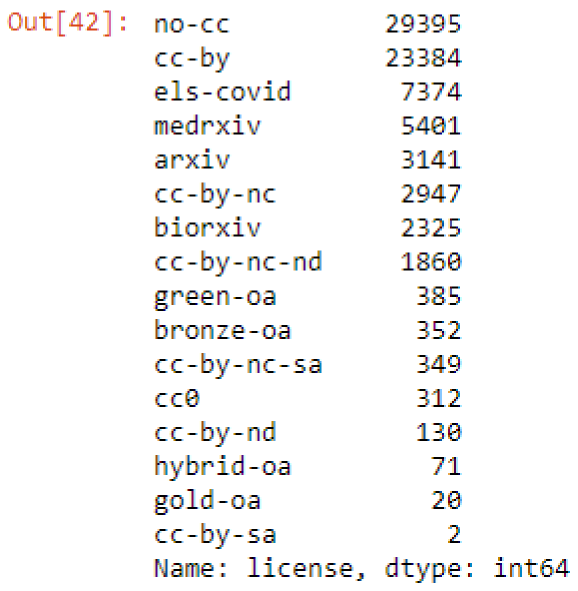


Figure 9: Accumulated count of sources in CORD19

Consequently, PMC and Medline are responsible for most of the publications. The less affiliated publishers are Elvesier, WHO, MedRxiv, ArXiv, BiRxiV. From a viewpoint of an observer, these findings can be used to formulate an indication of the CORD19 data origin. Indeed, it seems that the dominating sources for COVID related publications are PMC and Medline. Potential reasons for this indication could be that these publishers are focussing on medical research which boosts their appearance. Whereas the rest of the publishers may have their focus on ambiguous fields. Furthermore, the dominance of certain publishers can be linked to funding which means that dominating sources have more monetary power available than the smaller ones. Another possibility is linked to the algorithm outputting the software mentions. It could be easier to compute academic literature from PMC and Medline than from the less affiliated sources. As a result, the analysis of the column source_x lead to assumptions that could not be proved but enables the observer room for consideration.

7.1.5 Inspection of the column license

The analysis of the publication licenses is neglected due to its vague naming. Indeed, not all licenses could be identified properly. Furthermore, the column license does not have crucial importance for the subsequent analysis. Each license is solely affiliated with one publication. Due to completeness purposes, Figure 10 is incorporated into this dissertation to show all found licenses and their count.



```
Out[42]: no-cc          29395
         cc-by          23384
         els-covid       7374
         medrxiv         5401
         arxiv           3141
         cc-by-nc        2947
         biorxiv         2325
         cc-by-nc-nd     1860
         green-oa         385
         bronze-oa        352
         cc-by-nc-sa      349
         cc0              312
         cc-by-nd         130
         hybrid-oa         71
         gold-oa           20
         cc-by-sa           2
         Name: license, dtype: int64
```

Figure 10: Count of licenses in CORD19

7.1.6 Inspection of the column publish_time

Subsequently, the analysis of the column publish_time is conducted to display publishing periods. First, a pie chart is created to investigate the ratio between two periods. Therefore, the periods are divided into research that was published before 2020 and subsequently. Consequently, this distinction is beneficial as is it shows affiliation to COVID19. The transition between both years is chosen due to the occurring media attention during these days. In detail, the Chinese authorities have confirmed on 2019/31/12 that they were treating patients which were suffering from an unknown pneumonia disease. Several days later, Chinese researchers identified a virus that caused dozens of infections among people in Asia [45]. Based on the publication date, the dominance of COVID19 is shown in Figure 11.

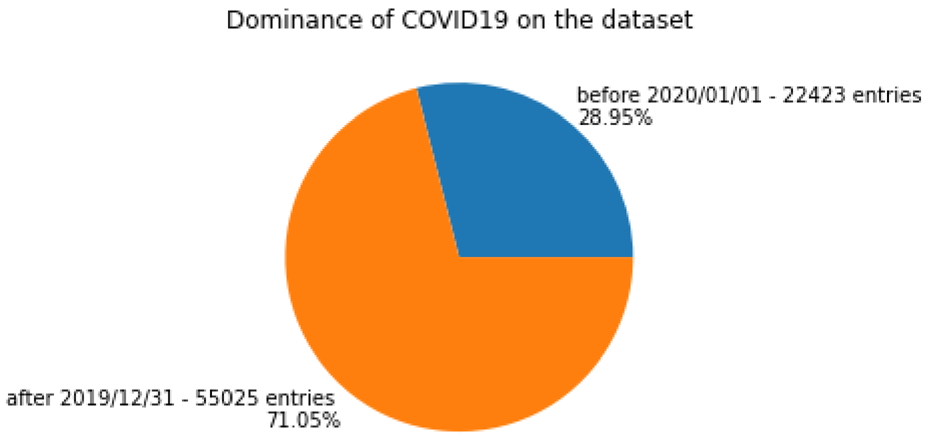


Figure 11: Pie chart - dominance of COVID19 publications

The figure shows that two-third of the dataset are assigned to COVID19 and one third was published priorly. This leads to potential assumptions that the latest Coronavirus outbreak was not the first one. Alternatively, one-third of the data is not inspecting COVID19 but other virus outbreaks. Thus, Figure 12 displays the entire publication period of the CORD19 dataset.

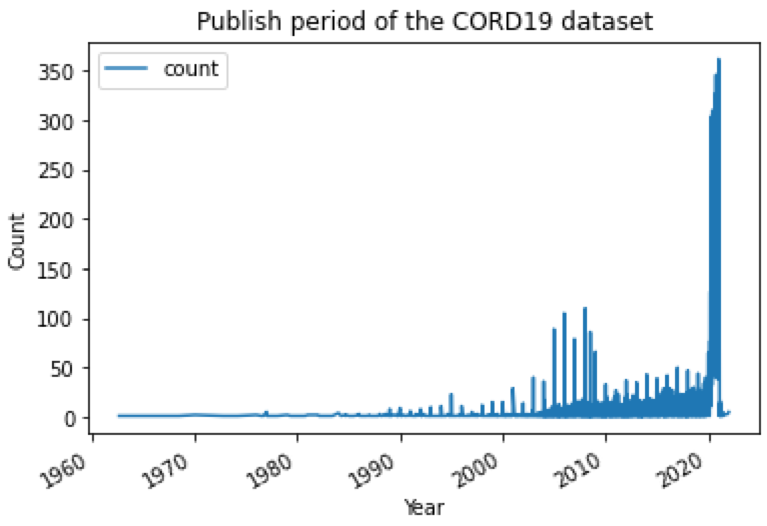


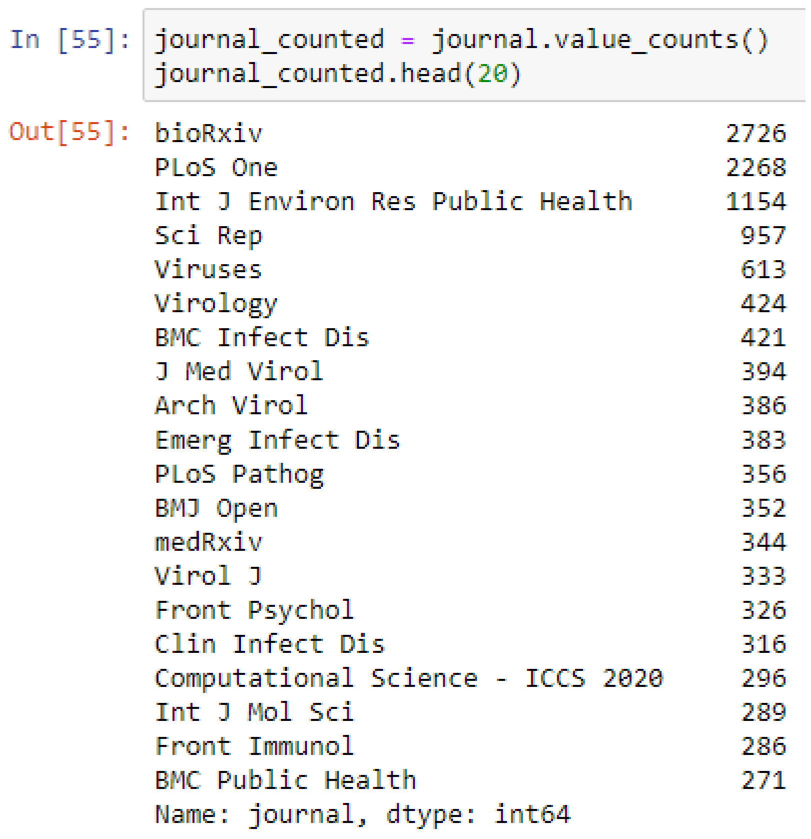
Figure 12: Line graph - CORD19 publication dates

Based on the figure above, it can be concluded that there were several Coronavirus outbreaks before 2020. Initially, this virus appeared in the 1960s within academic literature and rested three decades under the radar until the first deflection happened. The plot displays different swings with a varying extent which can be affiliated to past Severe Acute Respiratory Syndrome (SARS) outbreaks [46].

From an analytical viewpoint, the extent of the plot swings can be associated with the impact of occurring outbreaks. More widespread and global SARS outbreaks seem to be reflected with a high count of publications. Whereas smaller and regional events are affiliated with fewer publications. In terms of the dataset quality, it needs to be considered to rename the dataset. Due to its name “CORD19”, the dataset purports to be solely linked to the latest SARS outbreak named SARS-CoV-2 [47].

7.1.7 Inspection of the column `journal`

The analysis of the column `journal` is limited to the total numbers and each supplier. Due to its low expressiveness, the column cannot be properly analysed. Initially, this column contains thousands of NaNs by default. Moreover, the series object contains nearly 8000 different journals which show a steadily decreasing distribution. For completeness purposes, Figure 13 is implemented to the dissertation and presents the most common journals.

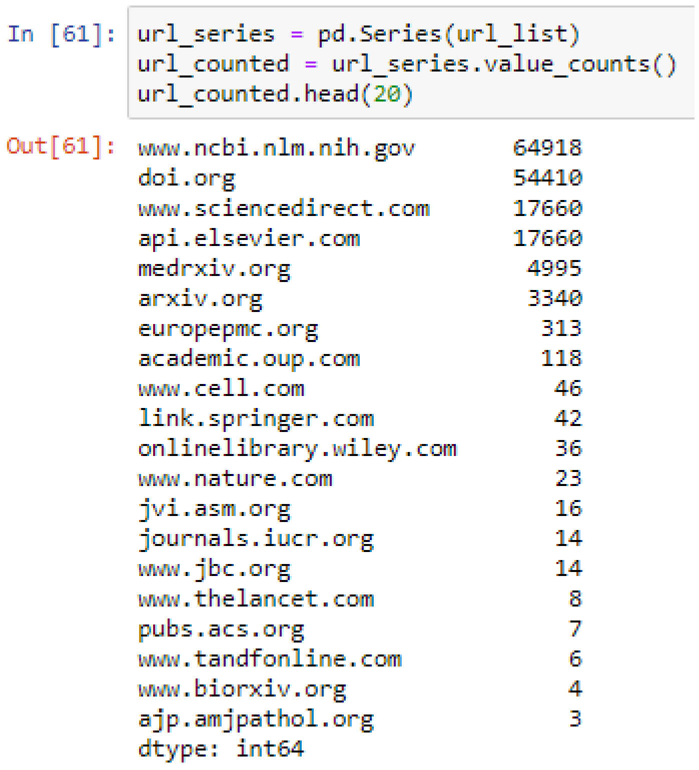


```
In [55]: journal_counted = journal.value_counts()
         journal_counted.head(20)

Out[55]: bioRxiv                              2726
         PLoS One                             2268
         Int J Environ Res Public Health      1154
         Sci Rep                               957
         Viruses                               613
         Virology                              424
         BMC Infect Dis                        421
         J Med Virol                           394
         Arch Virol                            386
         Emerg Infect Dis                      383
         PLoS Pathog                           356
         BMJ Open                              352
         medRxiv                               344
         Virol J                               333
         Front Psychol                         326
         Clin Infect Dis                       316
         Computational Science - ICCS 2020     296
         Int J Mol Sci                         289
         Front Immunol                         286
         BMC Public Health                     271
         Name: journal, dtype: int64
```

Figure 13: Count of journals in CORD19

7.1.8 Inspection of the column `url`

Based on the column `url`, the analysis investigates which hostnames and organisations contribute to which extent to this dataset. Usually, publishers host a website to provide academic literature. The column `url` contains one or more links pointing to the same publication. Consequently, some entries are published by various hostnames leading to an overlap of publishers for one entry. Thus, Figure 14 shows an extract of accumulated hostnames and reveals its distribution.



```
In [61]: url_series = pd.Series(url_list)
         url_counted = url_series.value_counts()
         url_counted.head(20)

Out[61]: www.ncbi.nlm.nih.gov        64918
         doi.org                     54410
         www.sciencedirect.com       17660
         api.elsevier.com            17660
         medrxiv.org                  4995
         arxiv.org                    3340
         europepmc.org                 313
         academic.oup.com              118
         www.cell.com                   46
         link.springer.com              42
         onlinelibrary.wiley.com        36
         www.nature.com                 23
         jvi.asm.org                    16
         journals.iucr.org              14
         www.jbc.org                    14
         www.thelancet.com               8
         pubs.acs.org                    7
         www.tandfonline.com             6
         www.biorxiv.org                 4
         ajp.amjpathol.org               3
         dtype: int64
```

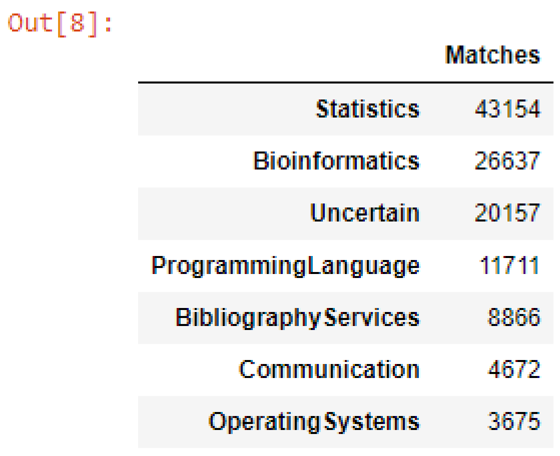
Figure 14: Count of hostnames in CORD19

From the figure above, it can be obtained that there are two providers which cover a vast portion of the publications. First, the `National Center for Biotechnology Information (NCBI)` is leading the table and facilitates the process of accessing genomic and biomedical information [48]. From an analytical view, the dominance of `NCBI` confirms that the dataset is excessively linked to the medical field. Nevertheless, the medical field is broad and the COVID related subjects depict a part of it. Secondly, the International `DOI Foundation (IDF)` is responsible for providing the second most publications. This organisation serves the registration for `doi` published research and is authorizing the procedure[49]. Due to its administrative role, the `IDF` is hosting a high number of academic work. These publications are not limited to a specific field which means that medical publications or distinct fields are likely to be part of the dataset. Subsequently, there are half a dozen providers which are responsible for most hosting activities. Whereas the remainder is serving a small number of academic research. From an analytical perspective, the dataset is robust towards potential shutdowns of hosting providers. Due to the circumstance that various providers host the same publication, information is more likely to persist. Nevertheless, this argument does not support all papers which indicates that there is academic literature that is solely hosted once.

7.2 Counting and categorisation of software mentions

The analysis of the column `software` is divided into two notebooks. First, the software mentions are accumulated with an algorithm flattening out duplicates and similar entries. Due to limited computational resources, the software counting process is fixed to a threshold and investigates a specific number of the most mentioned entries. This procedure is effective as it concentrates on the most influential software mentions. As a result, the leftover mentions have less impact on the potential ranking. Consequently, the implemented algorithm leads to a reduction of approximately 23.4% of mentions. The reduction is depending on the settings of the chosen fuzzy method that compares strings. From an analytical view, this notebook has less importance but is assigned more a supportive role which cleans data. Indeed, this notebook evolves from an unordered column with several software mentions per row to a DataFrame containing properly accumulated software mentions.

Subsequently, the second notebook is classifying the software mentions based on defined software categories. Therefore, the categories are created in a subjective manner by the author of this dissertation. Indeed, the author obeyed suggestions from the Habeas Corpus group to provide an insightful result that can be reused by the research group. Compared to the length of the dataset, it must be considered that a publication can be affiliated with more than one software mention. Thus, Figure 15 displays the outcome of the categorisation notebook.


Out[8]:

	Matches
Statistics	43154
Bioinformatics	26637
Uncertain	20157
ProgrammingLanguage	11711
BibliographyServices	8866
Communication	4672
OperatingSystems	3675

Figure 15: Classification of software mentions

The DataFrame and its given categories are a subjective product that outputs an individual outcome. Thus, the most used category within CORD19 is statistical software, followed by bioinformatics. Indeed, both categories are useful for the inspection of spreading viruses and analysing their genetic sequence. In the third position, there are mentions which could not be classified and are assigned to the category `Uncertain`. Due to the variety of the dataset, the classification caused the circumstance that entries tend to be unclassifiable. Moreover, there are entries with low expressiveness, or the software could not be researched for classification. The more specific mentions are assigned to `programming languages`, `bibliography services`, `communication`, and `operating systems`. For exploratory purposes, it must be clarified that the category `communication` can be described as a synonym to collaboration tools. Indeed, all categories can be considered as a synonym and rely on the subjective comprehension of observers.

Both notebooks handling the counting and classification of software mentions are more valuable in providing a reproducible code infrastructure rather than delivering an objective analysis. The classification process is linked to the intention of the author of this thesis to contribute to the Habeas Corpus group.

7.3 Exploration of SCOPUS information

The column `doi` was used to fetch affiliation data from the SCOPUS API. As a result, the API delivered information for 80.2 % of given `doi's`. This means that 19.8% of the data are not part of the subsequent analysis. Nevertheless, the analysis can be seen as representative because a large part of the data could be obtained from the SCOPUS API. Due to ethical reasons, the affiliated name of the institution and the retrieved publication city are not considered for the analysis. The affiliation data is limited to the country of the publication. Therefore, the entries per country are aggregated and visualized within a map which can be found in Figure 16.

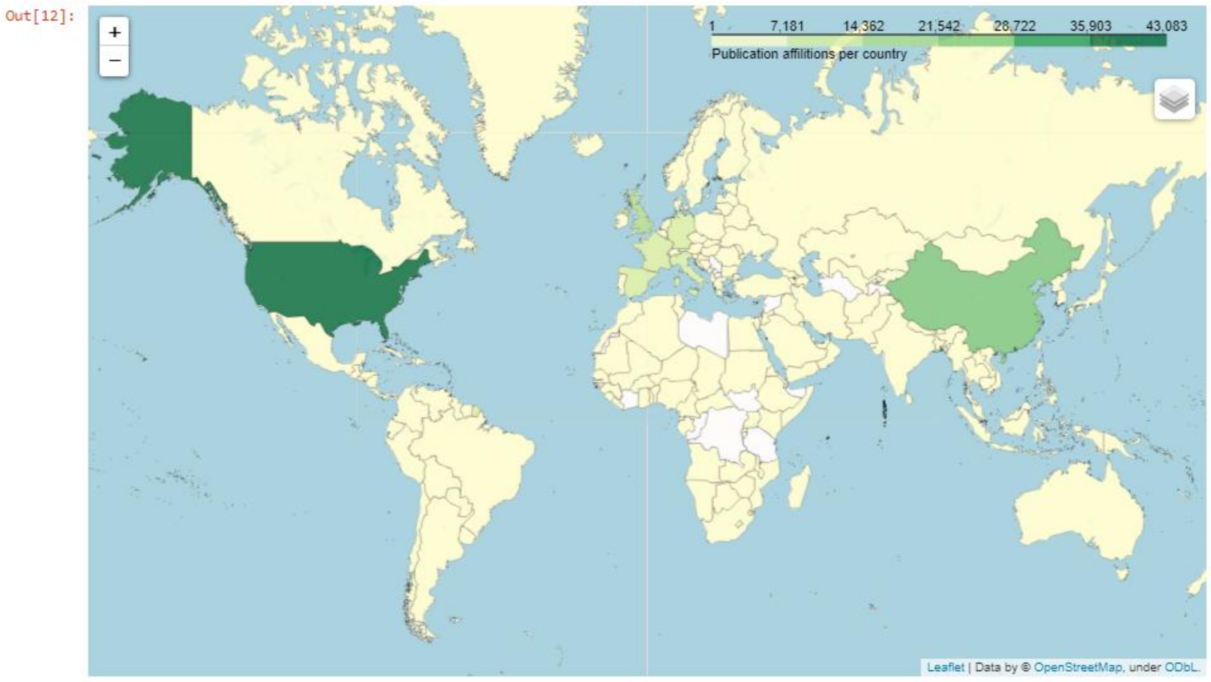

Out[12]:
+
−
1
7,181
14,362
21,542
28,722
35,903
43,083
Publication affilitions per country
Leaflet | Data by © OpenStreetMap, under ODbL.

Figure 16: Publication affiliations per country

The figure displays the affiliations per country correspondingly to a green colour hue. Countries with a lighter hue have fewer affiliations than regions with a darker tone. It can be obtained from the figure that nearly all countries have contributed to the dataset. There are some territories in Africa, in the Balkans and in the Middle East which did not research. On the world map, most of the areas are a contributor with a lighter colour hue. The dominating areas can be clearly distinguished by their darker hue which are the United States of America, the Republic of China, and numerous countries from Western Europe. From an analytical view, the most contributing countries are in an economically stable and wealthy position that can fund more research than less prospering ones. Furthermore, the population size or consternation by COVID is not linked to a countries capability to conduce to research.

Next to affiliation data, the SCOPUS API returned core data which contains details about the literature type and a citation count for each entry. Compared to the fetched affiliation data, the SCOPUS API simultaneously returned data for 81.2% of asked `doi's`. Due to this amount the analysis can be seen as representative. Moreover, the core data has a field that is critical as it violates the ethical framework of the thesis by describing author details. This information was removed before the analysis. Subsequently, Figure 17 displays a pie chart consisting of labels and counts for each literature type.

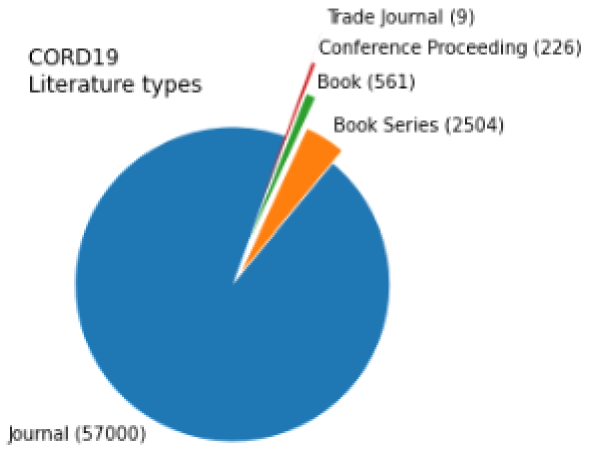


Figure 17: Literature types of CORD19 publications

The dominant literature type within the dataset are journals which are a vast majority of all entries. A smaller section of the dataset is affiliated to books and conference proceedings which still contribute to a recognizable number of publications. Trade journals are rarely distributed within the dataset. From an analytical viewpoint, journals might be the prevailing literature type as an author can publish this type more rapidly than books. Especially, a pandemic requests scientists to reveal their research in a short period to react flexibly to dynamic transforming circumstances.

Subsequently, the SCOPUS API delivers a citation count for each publication. For analytical purposes, it is significant to plot and inspect the pattern of the dataset which describes the number of citations per publication. Therefore, Figure 18 contains the number of publications on the vertical scale and its citation count on the horizontal axis.

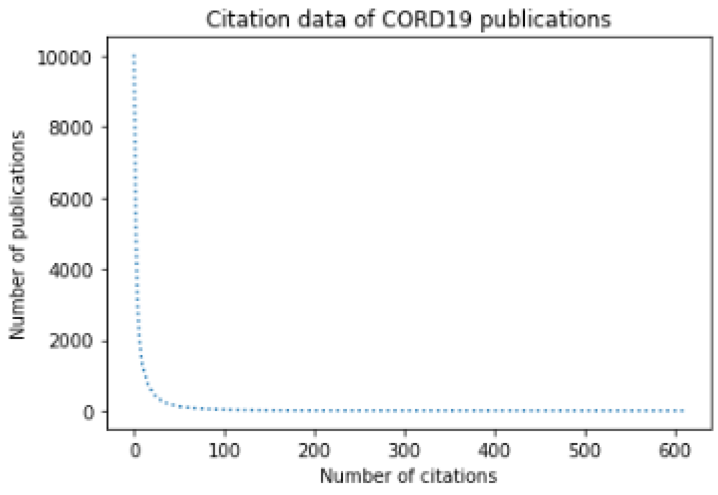


Figure 18: Citation data for CORD19 publications

The figure shows a typical logarithmic function with many publications having zero or a few citations. The number of citations is increasing for a decreasing number of academic research. From an analytical view, this indicates that most of the literature is recently published and available for the public. Alternatively, there is a selection of academic work which is monopolising attention and classifying the rest of the dataset as less relevant. Especially, the recent coronavirus outbreak led to spiking publication numbers which potentially pushed most academic work to the background.

8. Evaluation and critical appraisal

This project has successfully implemented a collection of Jupyter Notebooks which conduct an insightful and reproducible analysis of the CORD19 dataset. In terms of the defined objectives, the thesis followed and implemented all primary and secondary goals. Consequently, the research questions are answered thoroughly. Nevertheless, the secondary objective which intended to follow up some links from the dataset to external repositories was adapted to changing circumstances. For the thesis, it was planned to access the GitHub repositories which are affiliated to software mentions. For instance, GitHub holds information about software licenses that could be mined and analysed. Due to the poor naming specificity of software mentions, it is not possible to connect mentions to GitHub repositories. Instead, the column `doi` was used to fetch affiliation and other data from the SCOPUS API. The SCOPUS API was chosen due to its flexible search parameters and wide range of accessible data. Compared to the original objective of analysing licences linked to GitHub repositories, the analysis was conducted with alternative data provided by the SCOPUS API.

The external data provided insights linked to geolocation, literature type and citation count of publications. Furthermore, the dataset and its columns are explored individually to form findings. The count of software mentions is conducted with the implementation of Fuzzy-Wuzzy string comparison. Based on the aggregation and unification of software mentions a classification is provided. The corresponding notebooks conducting the count and classification of software mentions form a valuable structure that can be modified or reused. But the categorisation outcome needs to be considered cautiously due to its subjective design. With regards to the goals of the Habeas Corpus group, this analysis contributes to the target of this working group by generating insights. These insights are generated accordingly to the described related work and its guidelines. Especially, the introduced notebook principles are applied to a maximum extent enhancing code quality.

The success of this dissertation is founded on a work approach that can be described as a ping pong procedure between writing the thesis and implementing code. For this, the work amount for this thesis was divided into agile sprints which focus alternating on both parts of the project. First, academic literature is inspected and incorporated into the dissertation. Subsequently, the theoretical knowledge is applied to the notebooks performing an analysis. This approach is beneficial as it balances report and codebase. Moreover, it is facilitated to recognize relationships between textual parts and its code realisation.

Another work habit of this thesis is addressing solely the implementation. It was considered to “make the code work and then make it fast”. Consequently, the functionality of features was prioritised towards fast computing code. After each feature was available, the computation speed of the notebooks was improved. Due to the improvement of computation speed, the storage demand of the project decreased. Indeed, faster code resulted in a shift in data demand because notebooks focus solely on the most relevant data. Further reduction of storage was due to ethical guidelines which allowed to retrieve SCOPUS API data but restricted storage of personal identifying data such as author names and affiliations. Ultimately, the focus on relevant data is essential for performance and storage purposes.

To summarize, the project has successfully implemented a collection of notebooks that investigate the CORD19 dataset and can be applied to future versions of it. Therefore, the column structure of the dataset must persist. Furthermore, the notebooks fetching the SCOPUS API are usable for other datasets which contain `doi's`. Consequently, the generated findings from the automated as well as exploratory analysis posit an indicator for the performance of the dissertation.

9. Conclusions

This dissertation is accompanied by a collection of notebooks that extract various insights from the CORD19 dataset. Moreover, the notebooks provide a reproducible infrastructure for researchers that are interested in recreating the project. The key achievements of the project are findings that are established on a solid dataset quality consisting of representative columns.

Based on duplicates within columns that should solely include unique values, an improvement suggestion was revealed. It was proposed that the algorithm generating the dataset is required to check shared values and remove duplicates accordingly. Moreover, the analysis recognized that software mentions are not generated in a reproducible way. Indeed, two identical publications are not associated with identical software mentions. For this, it is essential to improve the responsible algorithm to output reproducible results. Further column dedicated analysis determined the affiliation to coronavirus-related topics. `PMC` and `Medline` are the main sources of the publications. The inspection of the publication dates leads to the outcome that one-third of the data is allocated to years before 2020. Consequently, the name CORD19 purports that the publications are solely linked to COVID19 which is not the case. For representative purposes, it needs to be considered to rename the dataset. The dominating hosting services are affiliated to `Elsevier`, `Sciencedirect`, `doi.org` and the `National Center for Biotechnology Information`. Consequently, statistics and bioinformatics are the most common software categories. Due to external data, additional findings could be created. From a viewpoint of regions, the United States of America, the Republic of China, and countries in Western Europe are core contributors to this dataset. The literature type `journal` is the most common, followed noticeable by books. Most of the publications have few or no citations, whereas a small selection is often cited.

During the development of notebooks, several drawbacks occurred which need to be clarified. The fetching duration of the SCOPUS API is limited to the capacity of external severs which means that it requires around four days to request all existing `doi's`. Moreover, the API does return information for four-fifth of `doi's`. It is connected to a database that is regularly updated which means that researchers need to frequently run the fetching notebook to fill up missing information. Especially, the citation count of publications is a rapid rising key number that needs to be constantly refreshed. Another disadvantage is linked to the notebook classifying software mentions. The classification process covers solely the most arising mentions and assigns subjective named categories. Due to the unspecific naming of mentions, it is challenging to manually categorize software mentions.

For future work, it must be considered to complete missing external data with other API services. Indeed, there is an `enrich_data()` functionality that concentrates on existing entries that could not be fetched. But the method is less likely to complete all the missing data. Furthermore, fetching Elsevier's database can be extended by searching for more features. Another issue addressing the accumulation of software mentions is the utilized Fuzzy-Wuzzy string comparison algorithm. Currently, the implemented threshold leads to a solid minimisation of 23.4% of the most common mentions. Nevertheless, the threshold can be optimized to increase reduction without diminishing the trade-off between the correct and incorrect accumulation of software mentions. For this, the heterogeneous naming of mentions must be challenged thoroughly. In general, all notebooks were optimized accordingly to the motto “make it work and then make it fast” but there is still room for improvement. In terms of computational speed, the notebooks processing fetched affiliation and core data are two chances for optimisation. Consequently, the described improvement approaches must consider additional possibilities to keep storage demand to a minimum. Indeed, future work is not limited to available recommendations. This project provides a foundation that can be utilized to search for new findings. After the submission and marking of this dissertation, the work will be continued in combination with a journal publication of the Habeas Corpus group. Finally, all findings were used to create insights and propose improvement proposals addressing the machine learning algorithm which extracted the CORD19 dataset.

References

[1] A. D. Wade and I. Williams, ‘Dryad Data -- CORD-19 Software Mentions’, 2021, Accessed: Jun. 01, 2021. [Online]. Available: https://doi.org/10.5061/dryad.vmcvdncs0.

[2] I. Beltagy, K. Lo, and A. Cohan, ‘SCIBERT: A pretrained language model for scientific text’, in *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, 2020, pp. 3615–3620, doi: 10.18653/v1/d19-1371.

[3] C. Du, J. Cohoon, P. Lopez, and J. Howison, ‘Softcite dataset: A dataset of software mentions in biomedical and economic research publications’, *J. Assoc. Inf. Sci. Technol.*, 2021, doi: 10.1002/asi.24454.

[4] V. Marx, ‘The big challenges of big data’, *Nat. 2013 4987453*, vol. 498, no. 7453, pp. 255–260, Jun. 2013, doi: 10.1038/498255a.

[5] A. M. Smith, D. S. Katz, and K. E. Niemeyer, ‘Software citation principles’, *PeerJ Comput. Sci.*, vol. 2016, no. 9, p. 86, Sep. 2016, doi: 10.7717/peerj-cs.86.

[6] Open Source Initiative, ‘History of the OSI | Open Source Initiative’, 2018. https://opensource.org/history (accessed Jun. 11, 2021).

[7] K. Fogel, *Producing open source software: How to run a successful free software project*. 2005.

[8] B. Perens and M. Sroka, ‘The Open Source Definition’, 2007. Accessed: Jun. 11, 2021. [Online]. Available: http://perens.com/OSD.html.

[9] E. Von Hippel and G. Von Krogh, ‘Open source software and the “private-collective” innovation model: Issues for organization science’, *Organ. Sci.*, vol. 14, no. 2, Apr. 2003, doi: 10.1287/orsc.14.2.209.14992.

[10] D. S. Katz *et al.*, ‘Software vs. data in the context of citation’, Dec. 2016, doi: 10.7287/peerj.preprints.2630v1.

[11] J. Münch, O. Armbrust, M. Kowalczyk, and M. Soto, *Software Process Definition and Management* . The Fraunhofer IESE Series on Software and Systems Engineering, 2012.

[12] IEEE, ‘ISO/IEC/IEEE 24765:2017(en), Systems and software engineering — Vocabulary’. https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:24765:ed-2:v1:en (accessed Jun. 12, 2021).

[13] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora, ‘Recovering traceability links in software artifact management systems using information retrieval methods’, *ACM Trans. Softw. Eng. Methodol.*, vol. 16, no. 4, Sep. 2007, doi: 10.1145/1276933.1276934.

[14] D. S. Katz *et al.*, ‘Software Citation Implementation Challenges’. Accessed: Jun. 14, 2021. [Online]. Available: https://arxiv.org/abs/1905.08674.

[15] eScience Lab at The University of Manchester, ‘Research Object Crate’. https://www.researchobject.org/ (accessed Jun. 15, 2021).

[16] H. Hata, J. L. C. Guo, R. G. Kula, and C. Treude, ‘Science-Software Linkage: The Challenges of Traceability between Scientific Knowledge and Software Artifacts’, Apr. 2021, Accessed: Jun. 08, 2021. [Online]. Available: http://arxiv.org/abs/2104.05891.

[17] R. Stojnic *et al.*, ‘The latest in Machine Learning | Papers With Code’. https://paperswithcode.com/ (accessed Jun. 08, 2021).

[18] D. Méndez Fernández, M. Monperrus, R. Feldt, and T. Zimmermann, ‘The open science initiative of the Empirical Software Engineering journal’, *Empirical Software Engineering*, vol. 24, no. 3. Springer New York LLC, pp. 1057–1060, Jun. 15, 2019, doi:

10.1007/s10664-019-09712-x.
[19] H. Hata, C. Treude, R. G. Kula, and T. Ishio, ‘9.6 Million Links in Source Code Comments: Purpose, Evolution, and Decay’, in *Proceedings - International Conference on Software Engineering*, May 2019, vol. 2019-May, pp. 1211–1221, doi: 10.1109/ICSE.2019.00123.
[20] N. Chue Hong, J. Cope, P. Herterich, D. S. Katz, and S. Worthington, ‘Recognising the value of software: how libraries can help the adoption of software citation’, Jun. 2021, doi: 10.6084/M9.FIGSHARE.14825268.V1.
[21] J. Howison and J. Bullard, ‘Software in the scientific literature: Problems with seeing, finding, and using software mentioned in the biology literature’, *J. Assoc. Inf. Sci. Technol.*, vol. 67, no. 9, pp. 2137–2155, Sep. 2016, doi: 10.1002/asi.23538.
[22] D. S. Katz and A. M. Smith, ‘Transitive Credit and JSON-LD’, *J. Open Res. Softw.*, vol. 3, no. 1, Nov. 2015, doi: 10.5334/jors.by.
[23] M. Jackson, ‘How to cite and describe software | Software Sustainability Institute’. https://www.software.ac.uk/how-cite-software (accessed Jun. 09, 2021).
[24] Y. H. Huang, P. W. Rose, and C. N. Hsu, ‘Citing a data repository: A case study of the Protein Data Bank’, *PLoS One*, vol. 10, no. 8, p. e0136631, Aug. 2015, doi: 10.1371/journal.pone.0136631.
[25] J. Starr *et al.*, ‘Achieving human and machine accessibility of cited data in scholarly publications’, *PeerJ Comput. Sci.*, vol. 2015, no. 5, p. e1, May 2015, doi: 10.7717/peerj-cs.1.
[26] M. K. Hensley, ‘Citation management software: Features and futures’, *Am. Libr. Assoc.*, vol. 50, no. 3, pp. 204–208, 2011, Accessed: Jun. 13, 2021. [Online]. Available: https://www.jstor.org/stable/41241164.
[27] I. GitHub, ‘About CITATION files - GitHub Docs’, 2021. https://docs.github.com/en/github/creating-cloning-and-archiving-repositories/creating-a-repository-on-github/about-citation-files (accessed Aug. 14, 2021).
[28] S. Druskat *et al.*, ‘Citation File Format’, Aug. 2021, doi: 10.5281/ZENODO.5171937.
[29] T. Kluyver *et al.*, ‘Jupyter Notebooks-a publishing format for reproducible computational workflows’, 2016, doi: 10.3233/978-1-61499-649-1-87.
[30] L. Yujian and L. Bo, ‘A normalized Levenshtein distance metric’, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1091–1095, Jun. 2007, doi: 10.1109/TPAMI.2007.1078.
[31] G. A. Rao, G. Srinivas, K. V. Rao, and P. V. G. D. P. Reddy, ‘A PARTIAL RATIO AND RATIO BASED FUZZY-WUZZY PROCEDURE FOR CHARACTERISTIC MINING OF MATHEMATICAL FORMULAS FROM DOCUMENTS’, *ICTACT J.*, vol. 8, no. 4, pp. 1728–1732, 2018, doi: 10.21917/ijsc.2018.0242.
[32] F. Javier Carrera Arias, ‘Fuzzy String Matching in Python’, 2019. https://www.datacamp.com/community/tutorials/fuzzy-string-python (accessed Jun. 24, 2021).
[33] Software Sustainability Institute, ‘About the Software Sustainability Institute ’. https://www.software.ac.uk/about (accessed Jun. 25, 2021).
[34] Software Sustainability Institute, ‘Manifesto ’. https://www.software.ac.uk/about/manifesto (accessed Jun. 25, 2021).
[35] Software Sustainability Institute, ‘GitHub - softwaresaved/habeas-corpus: A corpus of research software used in COVID-19 research.’ https://github.com/softwaresaved/habeas-corpus (accessed Jun. 25, 2021).
[36] A. Rule *et al.*, ‘Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks’, *PLoS Computational Biology*, vol. 15, no. 7. Public Library of

Science, p. e1007007, Jul. 01, 2019, doi: 10.1371/journal.pcbi.1007007.

[37] P. Guo, ‘Parse that data! Practical tips for preparing your raw data for analysis’, in *Perspectives on Data Science for Software Engineering*, Elsevier, 2016, pp. 169–173.

[38] the pandas development Team, ‘pandas.Series — pandas 1.2.5 documentation’. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.html (accessed Jul. 01, 2021).

[39] the pandas development Team, ‘pandas.Series.value_counts — pandas 1.2.5 documentation’. https://pandas.pydata.org/docs/reference/api/pandas.Series.value_counts.html (accessed Jul. 02, 2021).

[40] Elsevier, ‘Elsevier Developer Portal’, *Elsevier.com*. 2010, Accessed: Jul. 27, 2021. [Online]. Available: https://dev.elsevier.com/.

[41] Ale-de-vries, Ma-ji, Katrinleinweber, Adam3smith, and Matzech, ‘GitHub - ElsevierDev/elsapy: A Python module for use with Elsevier’s APIs: Scopus, ScienceDirect, others.’, Aug. 15, 2019. https://github.com/ElsevierDev/elsapy (accessed Jul. 27, 2021).

[42] Elsevier B.V., ‘Elsevier Developer Portal - Scopus Search Views’, 2021. https://dev.elsevier.com/sc_search_views.html (accessed Jul. 27, 2021).

[43] the Python Graph Gallery, ‘Choropleth map with Python and Folium’, 2018. https://www.python-graph-gallery.com/292-choropleth-map-with-folium (accessed Jul. 30, 2021).

[44] Rob Story, ‘Quickstart — Folium 0.12.1 documentation’, 2013. https://python-visualization.github.io/folium/quickstart.html (accessed Jul. 30, 2021).

[45] D. B. Taylor, ‘The Coronavirus Pandemic: A Timeline - The New York Times’, Mar. 17, 2021.

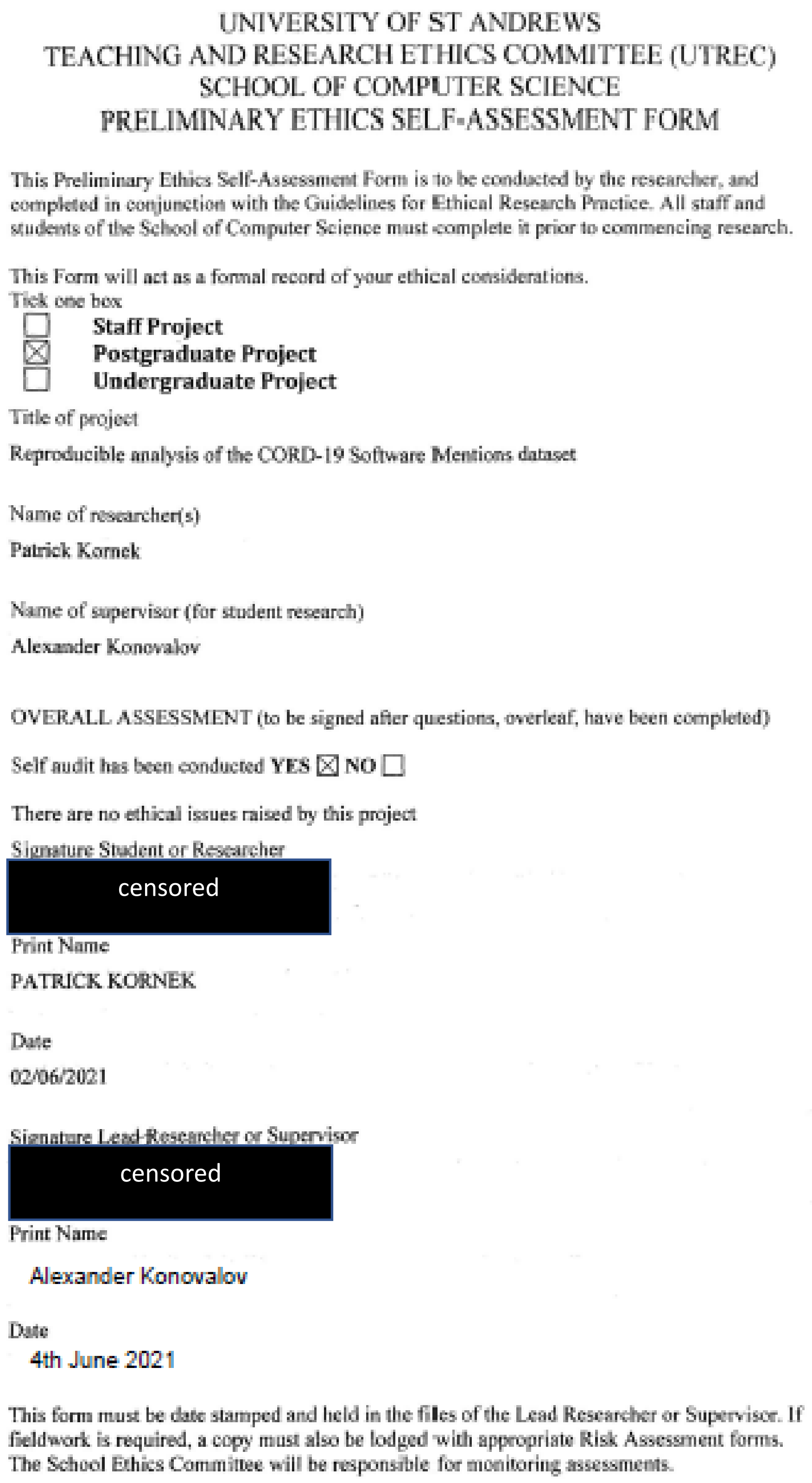
[46] U.S. Department of Health & Human Services, ‘CDC SARS Response Timeline | About | CDC’, Apr. 26, 2013. https://www.cdc.gov/about/history/sars/timeline.htm (accessed Jul. 20, 2021).

[47] J. A. Al-Tawfiq and A. J. Rodriguez-Morales, ‘Super-spreading events and contribution to transmission of MERS, SARS, and SARS-CoV-2 (COVID-19)’, *J. Hosp. Infect.*, vol. 105, no. 2, pp. 111–112, Jun. 2020, doi: 10.1016/J.JHIN.2020.04.002.

[48] National Center for Biotechnology Information, ‘National Center for Biotechnology Information’. https://www.ncbi.nlm.nih.gov/ (accessed Jul. 21, 2021).

[49] International DOI Foundation, ‘Digital Object Identifier System’, May 13, 2021. https://www.doi.org/ (accessed Jul. 21, 2021).

Appendix A - Preliminary Ethics Self-Assessment Form



UNIVERSITY OF ST ANDREWS
TEACHING AND RESEARCH ETHICS COMMITTEE (UTREC)
SCHOOL OF COMPUTER SCIENCE
PRELIMINARY ETHICS SELF-ASSESSMENT FORM

This Preliminary Ethics Self-Assessment Form is to be conducted by the researcher, and completed in conjunction with the Guidelines for Ethical Research Practice. All staff and students of the School of Computer Science must complete it prior to commencing research.

This Form will act as a formal record of your ethical considerations.
Tick one box

☐ **Staff Project**
[✓ **Postgraduate Project**]
☐ **Undergraduate Project**

Title of project

Reproducible analysis of the CORD-19 Software Mentions dataset

Name of researcher(s)

Patrick Kornek

Name of supervisor (for student research)

Alexander Konovalov

OVERALL ASSESSMENT (to be signed after questions, overleaf, have been completed)

Self audit has been conducted [✓ **YES**] **NO** ☐

There are no ethical issues raised by this project

Signature Student or Researcher

censored

Print Name

PATRICK KORNEK

Date

02/06/2021

Signature Lead Researcher or Supervisor

censored

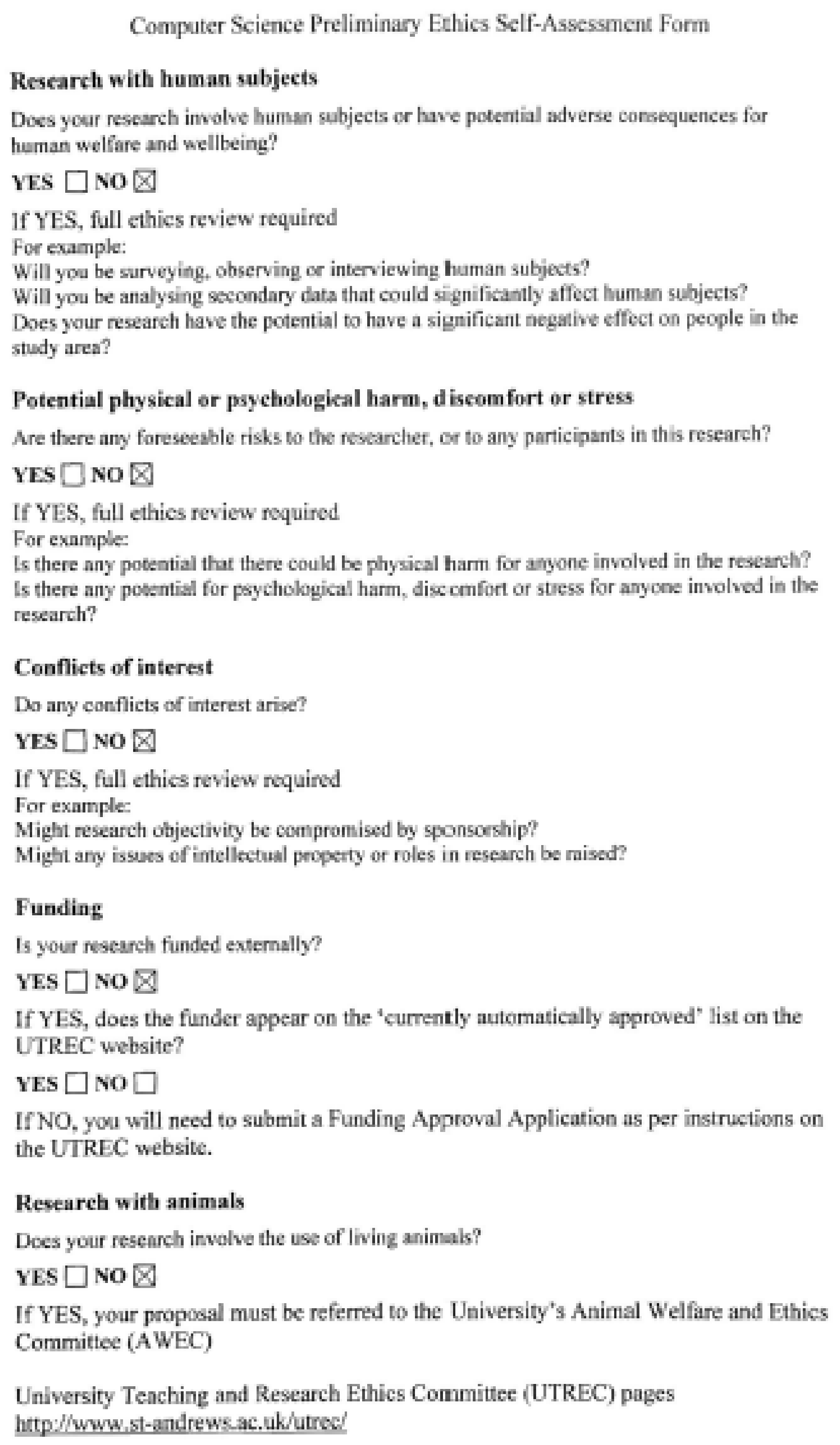
Print Name

Alexander Konovalov

Date

4th June 2021

This form must be date stamped and held in the files of the Lead Researcher or Supervisor. If fieldwork is required, a copy must also be lodged with appropriate Risk Assessment forms. The School Ethics Committee will be responsible for monitoring assessments.



Computer Science Preliminary Ethics Self-Assessment Form

Research with human subjects

Does your research involve human subjects or have potential adverse consequences for human welfare and wellbeing?

YES ☐ **NO** ☒

If YES, full ethics review required
For example:
Will you be surveying, observing or interviewing human subjects?
Will you be analysing secondary data that could significantly affect human subjects?
Does your research have the potential to have a significant negative effect on people in the study area?

Potential physical or psychological harm, discomfort or stress

Are there any foreseeable risks to the researcher, or to any participants in this research?

YES ☐ **NO** ☒

If YES, full ethics review required
For example:
Is there any potential that there could be physical harm for anyone involved in the research?
Is there any potential for psychological harm, discomfort or stress for anyone involved in the research?

Conflicts of interest

Do any conflicts of interest arise?

YES ☐ **NO** ☒

If YES, full ethics review required
For example:
Might research objectivity be compromised by sponsorship?
Might any issues of intellectual property or roles in research be raised?

Funding

Is your research funded externally?

YES ☐ **NO** ☒

If YES, does the funder appear on the 'currently automatically approved' list on the UTREC website?

YES ☐ **NO** ☐

If NO, you will need to submit a Funding Approval Application as per instructions on the UTREC website.

Research with animals

Does your research involve the use of living animals?

YES ☐ **NO** ☒

If YES, your proposal must be referred to the University's Animal Welfare and Ethics Committee (AWEC)

University Teaching and Research Ethics Committee (UTREC) pages
http://www.st-andrews.ac.uk/utrec/