Hochschule Wismar
University of Applied Sciences
Technology, Business and Design
Fakultät für Ingenieurwissenschaften

Master-Thesis

Identifikation und Vergleich von Autorenangaben zu Software zwischen verschiedenen Datenquellen

Eingereicht am:	24. September 2024
von:	Kevin Jahrens geboren am 05.08.1999 in Bad Oldesloe
Matrikelnummer:	480592
Betreuer:	Prof. Dr. -Ing. Frank Krüger Hochschule Wismar, Fakultät für Ingenieurwissenschaften Bereich Elektrotechnik und Informatik, Wismar, Deutschland
Zweitbetreuer:	M.A. Stephan Druskat Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR), Institut für Softwaretechnologie, Berlin, Deutschland

Hochschule Wismar
University of Applied Sciences
Technology, Business and Design
Fakultät für Ingenieurwissenschaften
Bereich Elektrotechnik und Informatik

Wismar, 1. Juli 2024

Master-Thesis

für: Herr **Kevin Jahrens**

Identifikation und Vergleich von Autorenangaben zu Software zwischen verschiedenen Datenquellen

---

Identifikation and comparison of authors of software across different data sources

---

Disposition

Software spielt eine zentrale Rolle in der Wissenschaft und sollte daher in wissenschaftlichen Arbeiten zitiert werden. Insbesondere für Autoren wissenschaftlicher Software ist die Zitation wesentlicher Bestandteil der wissenschaftlichen Anerkennung, sodass diese auch zunehmend in wissenschaftlichen Lebensläufen genannt werden und Beachtung finden. Anders als bei wissenschaftlichen Publikationen ist bei wissenschaftlicher Software aktuell noch unklar, welcher Anteil an der Entwicklung zu einer Nennung als Autor führt. Darüber hinaus existieren in verschiedenen Datenquellen widersprüchliche Angaben für Zitationsvorschläge bzgl. der Autoren einer Software.
Ziel dieser Masterarbeit ist es zu untersuchen inwieweit sich die Angaben von Autoren für Open Source Software unterscheiden. Dazu sollen öffentlich verfügbare Repositorien mit R und Python Paketen – als Stellvertreter für wissenschaftliche Software – hinsichtlich ihrer Autorenangaben untersucht werden. Insbesondere sollen die angegebenen Metadaten in den Repositorien (z.B. citation.cff) mit den Metadaten in Paketdatenbanken (`https://pypi.org/` und `https://cran.r-project.org/`) und den Entwicklungsanteilen automatisch verglichen werden.

1. Literaturrecherche Autorenrolle in Open Source Software und zur Disambiguierung von Autorennamen
2. Datensammlung: Identifikation und Download verfügbarer Metadaten zu „wichtigen“ Softwarepaketen
3. Automatische Auflösung und Abgleich der Autorennennungen aller Datenquellen
4. Analyse von Unterschieden in der Nennung von Autoren
5. Dokumentation der Ergebnisse in einer schriftlichen Master-Thesis

Startdatum: 16.09.2024
Abgabedatum: 17.03.2024

M. Litschke

Prof. Dr. rer. nat. Litschke
Chairman of the Examination Committee

Digitally signed by: Frank Krueger
Email: frank.krueger@hs-wismar.de
Date: 26.08.24

Frank Krüger

Prof. Dr.-Ing. Krüger
Supervisor

Abstract

Maximal eine halbe Seite.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

1.2 Vorgehen

1.3 Gliederung

2 Grundlagen

In Abschnitt 2.1 wird auf die Prinzipien der Software Zitation eingegangen. Es wird beschrieben, warum die Zitation von Software ebenfalls wichtig ist, ähnlich wie die Zitation von anderen wissenschaftlichen Arbeiten. Außerdem wird darauf eingegangen, dass ebenfalls Personen zitiert werden sollten, welche nicht aktiv an der Software Programmieren.

Autoren von Software werden in unterschiedlichen Quellen zitiert. Einige dieser Quellen sind stark mit der Softwareentwicklung verbunden. So gibt es verschiedene Systeme, die ein Entwickler verwenden kann, um seine Arbeit zu erleichtern bzw. überhaupt sinnvoll zu ermöglichen, in denen Sie anschließend als Autoren genannt werden. In den Abschnitten 2.2 und 2.3 wird auf die Versions- und Paketverwaltung eingegangen, welche zwei dieser Systeme darstellen. Des Weiteren existieren spezielle Zitierformate, in welchen Autoren explizit angegeben werden können. Auf diese Formate wird in Abschnitt 2.4 eingegangen. Außerdem können in Fließtexten, beispielsweise der Beschreibung einer Software, ebenfalls Autoren genannt werden. In Abschnitt 2.5 wird auf die *Named Entity Recognition* eingegangen, welche eine Methode darstellt, um Personen in Texten zu erkennen.

Alle Quellen, welche beschrieben werden dienen im Verlauf der Masterarbeit als Grundlage für die Extraktion von Autoren und deren Metainformationen. Die extrahierten Autoren müssen anschließend zugeordnet werden. Der Prozess dafür heißt *Author Name Disambiguation*, welcher in Abschnitt 2.6 beschrieben wird. Eine weitere einfache Möglichkeit des Abgleichs ist ein einfacher String Abgleich. Dieser funktioniert jedoch nicht immer, da Autoren unterschiedliche Schreibweisen ihres Namens verwenden können. Aus diesem Grund wird in Abschnitt 2.7 auf die Fuzzy-Suche eingegangen, welche eine Möglichkeit darstellt, um ähnliche Strings miteinander zu vergleichen, beispielsweise für den Abgleich von Namen mit oder ohne genannten Zwischennamen.

2.1 Zitation von Software

Software ist ein wesentlicher Bestandteil moderner Forschung. In der wissenschaftlichen Literatur ist es üblich, Quellen zu zitieren, um die Nachvollziehbarkeit und Reproduzierbarkeit von wissenschaftlichen Arbeiten zu gewährleisten. Im Gegensatz dazu ist dies bei wissenschaftlicher Software aktuell in diesem Umfang noch nicht gegeben. Hier gibt es aktuell kaum Anerkennung und Unterstützung für die Leistungen einzelner Autoren. Aus diesem Grund hat die „FORCE11 Software Zitier Arbeitsgruppe“ Prinzipien der Software Zitation erstellt, welche eine breite Akzeptanz in der wissenschaftlichen Gemeinschaft finden sollen. Im Folgenden werden die Prinzipien vorgestellt und erläutert Smith, Katz und Niemeyer 2016:

1. **Wichtigkeit:** Software sollte ein seriöses und zitierbares Produkt wissenschaftlicher Arbeit sein. Software Zitierungen sollten im wissenschaftlichen Kontext die gleiche Bedeutung zugeschrieben bekommen wie Zitierungen anderer Forschungsprodukte, wie Publikationen. Sie sollten wie Publikationen auch in der Arbeit enthalten sein, zum Beispiel in der Referenzliste eines Artikels. Software sollte auf derselben Grundlage zitiert werden wie jedes andere Forschungsprodukt auch, wie zum Beispiel ein Aufsatz oder ein Buch. Das bedeutet, dass Autoren die entsprechend verwendete Software zitieren sollten, so wie sie die entsprechenden Publikationen zitieren würden.
2. **Anerkennung und Zuschreibung:** Softwarezitate sollten die wissenschaftliche Anerkennung und die normative, rechtliche Würdigung aller Mitwirkenden an der Software ermöglichen, wobei anerkannt wird, dass ein einziger Stil oder ein Mechanismus für die Namensnennung nicht auf jede Software anwendbar sein kann.
3. **Eindeutige Identifikation:** Ein Softwarezitat sollte eine Methode zur Identifikation enthalten, die maschinell verwertbar, weltweit eindeutig und interoperabel ist und zumindest von einer Gemeinschaft der entsprechenden Fachleute und vorzugsweise von allgemeinen Forschern anerkannt wird.
4. **Persistenz:** Eindeutige Identifikatoren und Metadaten, die die Software und ihre Verwendung beschreiben, sollten bestehen bleiben – auch über die Lebensdauer der Software hinaus, die sie beschreiben.
5. **Zugänglichkeit:** Softwarezitate sollten den Zugang zur Software selbst und zu den zugehörigen Metadaten, Dokumentationen, Daten und anderen Materialien erleichtern, die sowohl für Menschen als auch für Maschinen notwendig sind, um die referenzierte Software sachkundig nutzen zu können.

6. **Spezifizität:** Softwarezitate sollten die Identifizierung und den Zugang zu der spezifischen Version der verwendeten Software erleichtern. Die Identifizierung der Software sollte so spezifisch wie nötig sein, z. B. durch Versionsnummern, Revisionsnummern oder Varianten wie Plattformen.

In dieser Arbeit wird verstärkt auf das Prinzip der Wichtigkeit eingegangen, da besonders im Citation File Format (CFF) und BibTEX Format die Möglichkeit besteht nicht die Software, sondern beispielsweise ein Artikel anzugeben. Diese Zitierweise würde dann das Prinzip der Wichtigkeit verletzen, da die Software nicht die gleiche Bedeutung zugeschrieben bekommt wie andere Forschungsprodukte. Diese Diskrepanz wird in Kapitel 4 dargestellt.

Es gibt verschiedene Gründe, warum die Zitation von Software ebenfalls wichtig ist und auch, dass Standards der Zitation eingehalten werden, ähnlich wie es der Fall bei anderen wissenschaftlichen Arbeiten ist. Einige dieser Gründe werden im Folgenden genannt (Smith, Katz und Niemeyer 2016):

- Forschungsfelder verstehen: Software ist ein Produkt der Forschung und wenn sie nicht zitiert wird, werden Lücken in der Aufzeichnung der Forschung über den Fortschritt in diesem Forschungsfeld entstehen.
- Anerkennung: Akademische Forscher auf allen Ebenen, einschließlich Studenten, Postdocs, Dozenten und Mitarbeiter, sollten für die Softwareprodukte, die sie entwickeln und zu denen sie beitragen, anerkannt werden, insbesondere wenn diese Produkte die Forschung anderer ermöglichen oder fördern. Nicht-akademische Forscher sollten ebenfalls für ihre Softwarearbeit anerkannt werden, obwohl die spezifischen Formen der Anerkennung sich von denen für akademische Forscher unterscheiden.
- Software entdecken: Mithilfe von Zitaten kann die in einem Forschungsprodukt verwendete Software gefunden werden. Weitere Forscher können dann dieselbe Software für andere Zwecke verwenden, was zu einer Anerkennung der für die Software Verantwortlichen führt.
- Reproduzierbarkeit: Die Angabe der verwendeten Software ist für die Reproduzierbarkeit notwendig, aber nicht ausreichend. Zusätzliche Informationen wie Konfigurationen und Probleme auf der Plattform sind ebenfalls erforderlich.

Wie bereits erwähnt werden Autoren in unterschiedlichen Quellen angegeben. Einige dieser Quellen werden in dieser Masterarbeit untersucht, wie beispielsweise das CFF und BibTEX Format. Die Autoren in diesen Quellen werden von dem jeweiligen Softwareprojekt angegeben. Falls an dem Projekt nicht nur Softwareentwickler

beteiligt sind, sondern beispielsweise auch Grafikdesigner oder Übersetzer, so sollten diese ebenfalls in den Quellen angegeben werden. Dies ist wichtig, da diese Personen ebenfalls einen Beitrag zum Projekt geleistet haben und somit auch Anerkennung verdienen.

Es gibt ebenfalls bereits Projekte, welche sich für die halb automatische Nennung von Autoren ohne Code Beitrag einsetzen. Ein Beispiel für ein solches Projekt ist „All Contributors“ (All Contributors 2024). Bei der Verwendung von „All Contributors“ werden die Mitwirkenden jedoch nicht in einem speziellen Format angegeben, sondern standardmäßig in der Readme-Datei, welche im Projekt enthalten ist. Die Liste wird dabei von einem Bot automatisch generiert und aktualisiert, sodass sie der Spezifikation von „All Contributors“ entspricht. Über einen Befehl in einem Issue oder einem Pull Request kann ein neuer Mitwirkender hinzugefügt werden. Autoren, welche keinen Code beigetragen haben, stellen im weiteren Verlauf der Masterarbeit ein Problem dar, da diese nicht mit den Autoren aus den Quellen abgeglichen werden können, da sie keine Commits haben. Dies wird in Abschnitt 3.2 genauer erläutert.

Die untersuchten Pakete in dieser Arbeit sind Open Source, welche auf GitHub veröffentlicht werden. Open Source Software ist Software, deren Quellcode öffentlich zugänglich ist und von einer Gemeinschaft von Entwicklern entwickelt wird. Die Entwickler arbeiten dabei in der Regel ehrenamtlich und ohne Bezahlung an der Software, wobei einige der untersuchten Pakete auch von Organisationen veröffentlicht werden, wie beispielsweise die „google-auth-library-python “ von Google. Diese wird primär von Google Mitarbeitern entwickelt und gepflegt. Wie bereits beschrieben sind Prinzipien und Gründe definiert, warum Software ebenfalls zitiert werden sollte. Welche Autoren in den Quellen angegeben werden sollten, ist jedoch nicht so genau definiert wie es beispielsweise im Bereich von wissenschaftlichen medizinischen Artikel der Fall ist. In diesem Bereich hat das „International Commitee of Medical Journal Editors“ Richtlinien für die Rolle von Autoren und Beitragenden in wissenschaftlichen Artikeln definiert (*ICMJE | Recommendations | Defining the Role of Authors and Contributors* 2024). Unter anderem aus diesem Grund ist die Menge der Autoren, welche in den Quellen angegeben werden, unterschiedlich und hängt von dem jeweiligen Projekt ab. Außerdem ist es möglich, dass ausschließlich Autoren in den Quellen angegeben werden, welche aktuell nicht mehr an dem Projekt beteiligt sind, aber beispielsweise vor fünf Jahren aktiv das Projekt geführt haben. Diese und weitere Auffälligkeiten werden in der Masterarbeit untersucht. Auch dies ist in anderen Bereichen anders definiert, sodass auch die neuen Autoren genannt werden müssen wie beispielsweise bei Neuauflagen von Büchern.

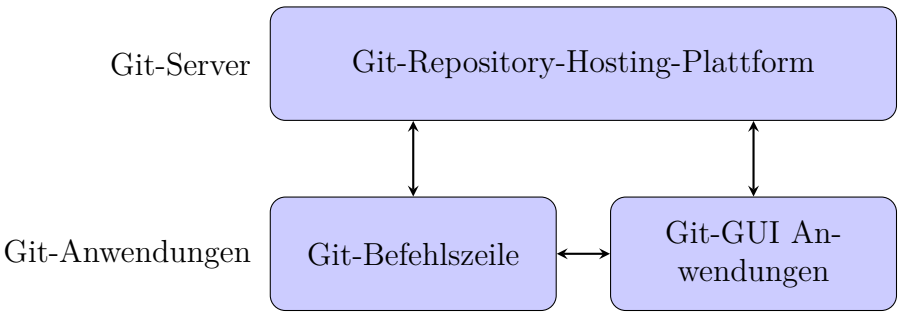


Abbildung 1: Übersicht über die Git-Komponenten
Die Git-Komponenten bestehen aus einem Git-Server, welcher das Repository hostet, und Git-Anwendungen, welche auf das Repository zugreifen (indirekt aus Ponuthorai und Loeliger 2022).

2.2 Versionsverwaltung

Die Versionsverwaltung ist ein System, um verschiedene Versionen von Software zu verwalten. Es bietet Zugang zu Code und dessen Änderungen in der Vergangenheit. Der Code und getätigte Änderungen werden in einem Repository gespeichert. Dadurch ist die Versionsverwaltung eine Art Logbuch, in dem alle Änderungen festgehalten werden. Dabei wird zusätzlich zu der Änderung der Autor und der Zeitpunkt der Änderung festgehalten (Ponuthorai und Loeliger 2022). Dies ermöglicht es in der Masterarbeit empirisch die Menge an Arbeit der einzelnen Autoren zu ermitteln.

Es gibt zwei verschiedene Arten von Versionsverwaltungssystemen. Zum einen gibt es die zentralen Systeme, bei denen alle Änderungen zentral verwaltet werden, beispielsweise SVN. Zum anderen gibt es die verteilten Systeme, bei denen jeder Entwickler eine Kopie des gesamten Repository und dessen Vergangenheit hat (ebd.). Ein solches System ist Git, welches sich mit einem Marktanteil von ungefähr 75 % gegenüber anderen Systemen durchgesetzt hat (Lindner 2024). Aus diesem Grund und weil Git-Repositorys in der Arbeit untersucht werden, wird auf Git eingegangen. Dabei werden Begriffe erklärt, mit denen es möglich ist, die geleistete Arbeit von einzelnen Autoren innerhalb eines Repositorys zu untersuchen. Der Aufbau der einzelnen Git Komponenten ist in Abbildung 1 dargestellt.

Bei der Benutzung von Git ist ein Server nicht zwingend erforderlich, jedoch steigert dies die Komplexität der Verwaltung und ist komplizierter in der Handhabung. Der Git-Server ermöglicht die einfache kollaborative Entwicklung von Code, da dieser ständig erreichbar ist und zentral verwaltet wird (Ponuthorai und Loeliger 2022). Standardmäßig wird auf dem Git-Server die neuste Version des Repositorys gespeichert. Ein möglicher Git-Server ist GitHub, auf welchen im späteren Verlauf weiter

eingegangen wird. Git-Anwendungen sind Programme, welche mit dem lokalen Repository interagieren (Ponuthorai und Loeliger 2022). Diese können auf entfernte Git-Repositorys zugreifen wie beispielsweise Repositorys, welche auf GitHub gehostet werden. Anschließend arbeiten die Programme auf der lokalen Kopie und können die Änderungen, wenn nötig, auf das entfernte Repository übertragen.

In Repositorys werden verschiedene Arten von Statistiken gespeichert. Git verwaltet Revisionen als *Snapshot*. Anders als in anderen Systemen wird keine Serie von Änderungen gespeichert, sondern ein *Snapshot* der Änderungen zu einem bestimmten Zeitpunkt erstellt (ebd.). Dies wird ein Commit genannt. An einem Commit werden verschiedene Metainformationen gespeichert. Unter anderem wird eine Commit-Nachricht, der Autor und der Zeitpunkt der Änderungen gespeichert. Mehrere Commits bilden die Commit-Historie bzw. die Vergangenheit eines Repositorys. Weitere Eigenschaften, welche sich aus dem Repository exportieren lassen, sind die Anzahl der eingefügten und gelöschten Zeilen. Außerdem lässt sich die Anzahl der geänderten Dateien ermitteln. Diese Werte können für das gesamte Repository oder für einzelne Autoren ermittelt werden.

Ein Repository kann verschiedene Branches enthalten, muss jedoch mindestens einen enthalten. In der Vergangenheit wurde der Hauptbranch *master* genannt. Seit 2020 wird dieser jedoch in *main* umbenannt, um rassistische Konnotationen zu vermeiden (GitHub 2024b). Ein Branch ist eine separate Entwicklungslinie, welche unabhängig von anderen Branches ist. Beim Erstellen von einem Branch wird der aktuelle Zustand des Branches, auf welchem der neue Branch erstellt wird, kopiert (Ponuthorai und Loeliger 2022). Dadurch können Änderungen in dem neuen Branch durchgeführt werden, ohne dass diese Änderungen den ursprünglichen Branch beeinflussen. Diese Änderungen werden mittels Commits festgehalten. Unterschiedliche Branches können anschließend zusammengeführt werden, um die Änderungen in einem Branch in einen anderen Branch zu übernehmen.

Die Statistiken der Repositorys können auf verschiedene Arten aufgearbeitet werden. Zum einen können einige direkt mittels Git-Befehlen ausgelesen werden (Chacon 2024). Andere wiederum benötigen komplexere Abfragen, welche beispielsweise mittels Skripten oder speziellen Programmen ausgelesen werden können. Ein Beispiel für ein Programm, welches Git-Statistiken aufarbeitet, ist *git-quick-stats* (Meštan 2024). Außerdem bieten Onlinedienste zur Versionsverwaltung, wie GitHub, Statistiken über APIs an, welche jedoch im Umfang der Anfragen limitiert sind (GitHub 2022).

GitHub ist eine Plattform, auf welcher Git-Repositorys gehostet werden können und dient somit als ein Git-Server. GitHub bietet zusätzliche Funktionen an, welche über die Standardfunktionen von Git hinausgehen. Diese umfassen unter anderem die kollaborative Entwicklung von Code, Automatisation mittels CI/CD, Sicherheitsaspekte, Projekt Management, Team Administration und Client-Anwendungen zur Verwaltung von Repositorys (Ponuthorai und Loeliger 2022). Aktuell benutzen GitHub über 100 Millionen Entwickler und mehr als 4 Millionen Organisationen. Insgesamt verwaltet die Plattform über 420 Millionen Repositorys. Außerdem ist GitHub in 90 % der Fortune 100 Unternehmen im Einsatz (GitHub 2024a). Um die zusätzlichen Funktionen von GitHub bereitzustellen werden sogenannte Issues, Pull Requests, geschützte Branches, Actions, Diskussionen und Wikis eingesetzt. GitHub Issues sind eine Möglichkeit, um Probleme und Aufgaben zu verfolgen. Pull Requests dienen dazu Änderungen in einem Branch eines Repositorys anzufragen und über diese zu informieren. In dem Pull Requests kann der Code überprüft und diskutiert werden.

2.3 Paketverwaltung

Im Gegensatz zur Versionsverwaltung verwaltet die Paketverwaltung keinen Code und dessen Änderungen, sondern fertige Softwarepakete, welche von Entwicklern erstellt und in einem Repository abgelegt werden. Inhalt eines Pakets können beispielsweise standardisierter Code von Software Modulen sein oder kompilierter Code. Zusätzlich werden in einem Paket Metadaten gespeichert. Diese Metadaten können beispielsweise eine Beschreibung, Version, Abhängigkeiten und Autoren des Paketes enthalten. Sie lassen sich aus dem Paket mithilfe des Paketverwaltungssystems auslesen. Außerdem übernimmt das Paketverwaltungssystem das Installieren und meistens auch das Aktualisieren und Deinstallieren von Paketen. Zusätzlich wird das System verwendet, um fehlende Abhängigkeiten von Paketen automatisch zu installieren (Spinellis 2012).

In dieser Arbeit wird auf zwei Paketverwaltungssysteme eingegangen. Zum einen wird auf PyPi eingegangen, welches das Paketverwaltungssystem für Python ist. Zum anderen wird auf CRAN eingegangen, welches das Paketverwaltungssystem für R ist. In PyPi sind aktuell mehr als 500.000 unterschiedliche Projekte mit über 5 Millionen Veröffentlichungen verfügbar (Python Software Foundation 2024a). Im Gegensatz dazu sind in CRAN aktuell mehr als 20.000 Pakete verfügbar (CRAN Team 2024).

PyPi stellt eine JSON API zur Verfügung, um die Metadaten einzelner Pakete abzufragen. Sie ist nicht in der Anzahl der Anfragen beschränkt (Python Software Foundation 2024b). Zusätzlich zur API werden auf der Webseite von PyPi verifizierte Owner und Betreuer der Pakete angezeigt, welche nicht über die API abgefragt werden können. Ebenfalls bietet PyPi über Google BigQuery einen Datensatz an, in denen sämtliche Pakete mit ihren Versionen und Metadaten enthalten sind.

CRAN selbst bietet keine API an, um die Metadaten der Pakete abzufragen. Jedoch gibt es das METACRAN-Projekt, welches eine Kollektion von kleinen Diensten für das CRAN-Repository bereitstellt. Eines dieser Dienste ist eine CouchDB, welche die Metadaten aller Pakete von CRAN bereitstellt. Eine CouchDB ist eine Apache Datenbank, welche nativ eine HTTP/JSON API bereitstellt (The Apache Software Foundation 2024). Die Datenbank ist eine Kopie des CRAN-Repository und wird regelmäßig aktualisiert (Csárdi und Salmon 2023). Die Ausgabe der API erfolgt in JSON und teilweise sind einzelne Felder in R formatiert.

2.4 Zitierformate

2.4.1 Citation File Format

2.4.2 BibTeX

2.5 Named Entity Recognition

2.6 Disambiguierung von Autorennamen

2.7 Fuzzy suche

3 Methodik

3.1 Datenbeschaffung

3.1.1 Git

3.1.2 PyPi

3.1.3 CRAN

3.1.4 Beschreibung

3.1.5 Citation File Format

3.1.6 BibTeX

3.2 Abgleich

4 Ergebnisse

5 Diskussion

5.1 Limitierungen

6 Fazit und Ausblick

6.1 Fazit

6.2 Ausblick

A Beispielanlage

Beispieltext.

Literaturverzeichnis

All Contributors (2024). *Recognize all contributors*. All Contributors. URL: https://allcontributors.org (besucht am 01. 06. 2024).

Chacon, Scott (2024). *Git - git-shortlog Documentation*. URL: https://git-scm.com/docs/git-shortlog (besucht am 21. 05. 2024).

CRAN Team (Mai 2024). *The Comprehensive R Archive Network*. URL: https://cran.r-project.org/ (besucht am 21. 05. 2024).

Csárdi, Gábor und Maëlle Salmon (2023). *pkgsearch: Search and Query CRAN R Packages*. URL: https://github.com/r-hub/pkgsearch.

GitHub (28. Nov. 2022). *Rate limits for the REST API*. GitHub Docs. URL: https://docs.github.com/en/rest/using-the-rest-api/rate-limits-for-the-rest-api?apiVersion=2022-11-28 (besucht am 21. 05. 2024).

– (2024a). *About*. Let's build from here. URL: https://github.com/about (besucht am 23. 09. 2024).

– (20. Sep. 2024b). *github/renaming*. original-date: 2020-06-12T21:51:22Z. URL: https://github.com/github/renaming (besucht am 24. 09. 2024).

ICMJE | Recommendations | Defining the Role of Authors and Contributors (2024). URL: https://www.icmje.org/recommendations/browse/roles-and-responsibilities/defining-the-role-of-authors-and-contributors.html (besucht am 21. 09. 2024).

Lindner, Jannik (3. Mai 2024). *Version Control Systems Industry Statistics*. URL: https://worldmetrics.org/version-control-systems-industry-statistics/ (besucht am 21. 05. 2024).

Mešťan, Lukáš (18. Mai 2024). *git-quick-stats*. Version 2.5.6. URL: https://github.com/arzzen/git-quick-stats (besucht am 21. 05. 2024).

Ponuthorai, Prem Kumar und Jon Loeliger (Nov. 2022). *Version Control with Git*. 3. Aufl. Sebastopol: O'Reilly Media. ISBN: 978-1-4920-9119-6.

Python Software Foundation (Mai 2024a). *PyPI · Der Python Package Index*. PyPI. URL: https://pypi.org/ (besucht am 21. 05. 2024).

– (Mai 2024b). *Warehouse documentation*. URL: https://warehouse.pypa.io/index.html (besucht am 21. 05. 2024).

Smith, Arfon M., Daniel S. Katz und Kyle E. Niemeyer (19. Sep. 2016). „Software citation principles". In: *PeerJ Computer Science* 2. Publisher: PeerJ Inc., e86. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.86. URL: https://peerj.com/articles/cs-86 (besucht am 10. 09. 2024).

Spinellis, Diomidis (2012). „Package Management Systems". In: *IEEE Software* 29.2, S. 84–86. DOI: 10.1109/MS.2012.38.

The Apache Software Foundation (2024). *Apache CouchDB*. URL: https://couchdb.apache.org/ (besucht am 22. 05. 2024).

Abbildungsverzeichnis

Tabellenverzeichnis

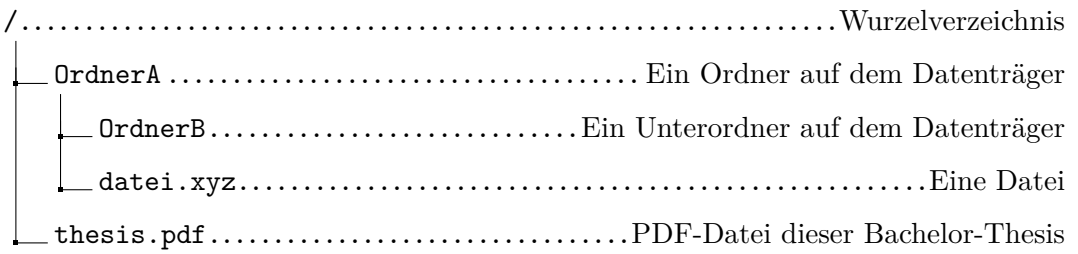
Algorithmenverzeichnis

Quellcodeverzeichnis

Abkürzungsverzeichnis

CFF Citation File Format. 8

Datenträger



Im Unterverzeichnis `tools` des Projekts findet sich das Perl-Skript `dirtree.pl`, mit welchem Inhalte für das dirtree-Environment (siehe oberhalb) semiautomatisch erstellt werden können.

Die Nutzung aus der Kommandozeile ist wie folgt:
`perl dirtree.pl /path/to/top/of/dirtree`

Quelle des Skripts:
`https://texblog.org/2012/08/07/semi-automatic-directory-tree-in-latex/`

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die anderen Quellen im Wortlaut oder dem Sinn nach entnommen wurden, sind durch Angaben der Herkunft kenntlich gemacht. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet.

Ich erkläre ferner, dass ich die vorliegende Arbeit in keinem anderen Prüfungsverfahren als Prüfungsarbeit eingereicht habe oder einreichen werde.

Die eingereichte schriftliche Arbeit entspricht der elektronischen Fassung. Ich stimme zu, dass eine elektronische Kopie gefertigt und gespeichert werden darf, um eine Überprüfung mittels Anti-Plagiatssoftware zu ermöglichen.

Wismar, den 24. September 2024
Ort, Datum Unterschrift