

# Identifikation und Vergleich von Autorenangaben zu Software zwischen verschiedenen Datenquellen

Wismar, 30. Januar 2025

Fakultät für Ingenieurwissenschaften, Hochschule Wismar

Kevin Jahrens

E-Mail: [k.jahrens@stud.hs-wismar.de](mailto:k.jahrens@stud.hs-wismar.de)



# Einleitung

# Motivation

- Software spielt zentrale Rolle in der Wissenschaft
- Zitation wesentlicher Bestandteil in wissenschaftlicher Publikation
- Bei wissenschaftlicher Software ist dies in diesem Umfang aktuell nicht gegeben
- Softwareautoren werden nicht immer genannt und manchmal sogar ihrer Beiträge beraubt [1]

# Vorgehen

- Autoren aus unterschiedlichen Quellen extrahieren
- Autoren untereinander abgleichen
- Ausschließliche Betrachtung von Autoren, die Code in Git beigetragen haben
- Ergebnisse aufbereiten
- Beantwortung von Forschungsfragen

# Forschungsfragen

- F1** Wie gut können Autoren untereinander abgeglichen werden?
- F2** Was muss ein Softwareentwickler leisten, um als Autor genannt zu werden?
- F3** Wie gut werden Autoren in den einzelnen Quellen gepflegt?

# Grundlagen

# Prinzipien der Software-Zitation [2]

1. **Wichtigkeit:** Software sollte ein seriöses und zitierbares Produkt wissenschaftlicher Arbeit sein.
2. **Anerkennung und Zuschreibung:** Softwarezitate sollten die wissenschaftliche Anerkennung und die normative, rechtliche Würdigung aller Mitwirkenden an der Software ermöglichen.
3. **Eindeutige Identifikation:** Ein Softwarezitat sollte eine Methode zur Identifikation enthalten, die maschinell verwertbar, weltweit eindeutig und interoperabel ist.
4. **Persistenz:** Eindeutige Identifikatoren und Metadaten, die die Software und ihre Verwendung beschreiben, sollten bestehen bleiben – auch über die Lebensdauer der Software hinaus.
5. **Zugänglichkeit:** Softwarezitate sollten den Zugang zur Software selbst und weiteren Materialien erleichtern, um sie sachkundig nutzen zu können.
6. **Spezifizität:** Softwarezitate sollten die Identifikation und den Zugang zu der spezifischen Version der verwendeten Software erleichtern. Die Identifizierung der Software sollte so spezifisch wie nötig sein.

# Versionsverwaltung


- Verwaltet Quellcode und dessen Änderungen in einem Repository
- Git ist eine weit verbreitete Versionsverwaltung mit einem Marktanteil von ungefähr 75 % [3]
- Speichert Zeitpunkt und Autor, sowie die Änderungen in einem Commit
- Name und E-Mail des Autors frei wählbar
- In Git werden weitere Daten gespeichert, welche ausgelesen werden können:
  - Anzahl der eingefügten und gelöschten Zeilen
  - Anzahl der geänderten Dateien
  - Anzahl der Commits
- Repositories können auf einem Server verwaltet werden
- Weit verbreiteter Anbieter eines Git-Servers ist GitHub



# Software-Verzeichnisse und Paketverwaltung

- Eine Paketverwaltung verwaltet fertige Softwarepakete, bspw. kompilierten Code
- Softwarepakete können in einem Software-Verzeichnis abgelegt werden
- Softwarepakete enthalten Metadaten, bspw. die Autoren des Pakets
- Es werden die Verzeichnisse PyPI (Python) und CRAN (R) untersucht
- Für beide Verzeichnisse stehen APIs zur Verfügung, welche die Metadaten bereitstellen

# Zitierformat – Citation File Format

- Wird als Datei z. B. in einem Git-Repository gespeichert
- Auf GitHub  verwaltet
- 2.512 Repositorys auf GitHub haben eine CFF-Datei (Stand 07.11.2024)
- Dient dazu anderen die Zitation der Software zu erleichtern
- Kann unter anderem durch Programme wie *cffinit* erstellt werden [4]
- *cffconvert* bietet die Möglichkeit der Validation und der Umwandlung z. B. in BibT<sub>E</sub>X [5]

```
1  cff-version: 1.2.0
2  title: "CompAuthorsBetweenDS"
3  message: "If you use this software, please cite it
   ↪ using the metadata from this file."
4  type: software
5  authors:
6    - given-names: "Kevin"
7      family-names: "Jahrens"
8      email: "k.jahrens@stud.hs-wismar.de"
9    - name: "Hochschule Wismar"
10 preferred-citation:
11   title: "Identifikation und Vergleich von
   ↪ Autorenangaben zu Software zwischen
   ↪ verschiedenen Datenquellen"
12 type: thesis
13 year: 2025
14 authors:
15   - given-names: "Kevin"
16     family-names: "Jahrens"
17     email: "k.jahrens@stud.hs-wismar.de"
18   - name: "Hochschule Wismar"
```

# Named Entity Recognition

- Beschreibt den Prozess der automatischen Erkennung und Klasseneinteilung von Substantiven (Entitäten) im Text
- Typische Entitäten sind Personen, Orte oder Organisationen
- Viele Anwendungsgebiete bspw. Informationsextraktion aus Texten

## Pytorch README

**PyTorch** (Organisation) *is currently maintained by* **Soumith Chintala** (Person), **Gregory Chanan** (Person), **Dmytro Dzhulgakov** (Person), **Edward Yang** (Person), and **Nikita Shulga** (Person) *with major contributions coming from* **hundreds** (Digit) *of talented individuals in various forms and means.*

# Unscharfe Suche

- Findet ähnliche Zeichenfolgen, die sich in ihrer Schreibweise unterscheiden
- Als Ergebnis wird zumeist eine Distanz zwischen zwei Zeichenfolgen in Prozent angegeben
- Viele Anwendungsgebiete bspw. die Suche eines Namens in einem Index

## Unscharfe Suche

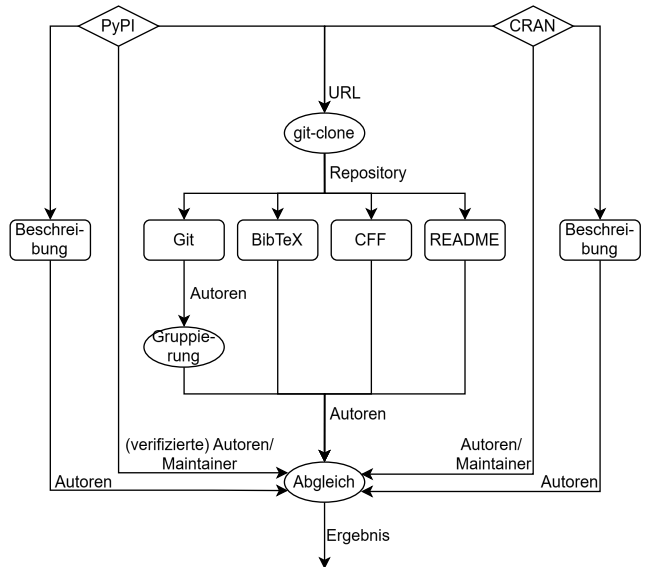
Soumith Chintala ↔ Soumith **S.** Chintala: 91 %

**Sou**mith Chintala ↔ **Su**omith Chintala: 94 %

# Methodik

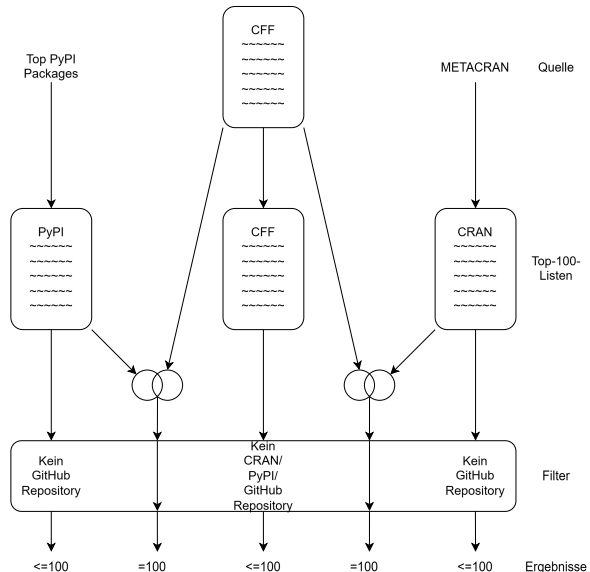
# Datenbeschaffung

- Datenbeschaffung für jeweils ein Paket (PyPI oder CRAN)
- Git, BibTeX, CFF und README werden zu Änderungszeitpunkten beschafft
- Beschreibung und README setzt NER ein



# Top-100-Listen

- 5 Top-100-Listen wurden erstellt und untersucht:
  1. Top-100 PyPI
  2. Top-100 CRAN
  3. Top-100 CFF
  4. Top-100 PyPI CFF
  5. Top-100 CRAN CFF
- Top-100 definiert an der Anzahl der Downloads (PyPI, CRAN) bzw. Sterne (CFF) auf GitHub



# Abgleich

- Abgleich jeweils zwischen Git und einer weiteren Quelle
- Vereinfachung: Abgleich der Git-Autoren des aktuellen Stands mit den Quellen
- Abgleich verwendet Python *in* und die unscharfe Suche
- Abgleich erfolgt anhand:
  - Name
  - E-Mail
  - Benutzername
- Für jeden Vergleich zweier Autoren wird ein Score berechnet
- Die Kombination mit dem besten Score wird ausgewählt
- Autoren mit mehr Commits werden bevorzugt



# Ergebnisse

# Abgleich

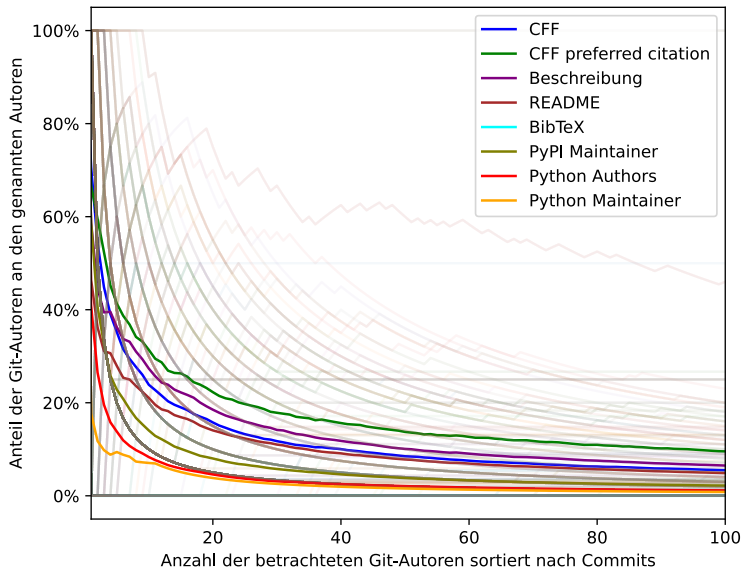
- Ergebnisse sind abhängig vom Abgleich
- Manuelle Prüfung von jeweils 2 Autoren jeder Quelle jedes Pakets
- Einteilung in:
  - **TP**: richtig als positiv klassifiziert
  - **FN**: fälschlicherweise als negativ klassifiziert (obwohl positiv)
  - **FP**: fälschlicherweise als positiv klassifiziert (obwohl negativ)
  - **TN**: richtig als negativ klassifiziert
- Berechnung des F1-Scores, ein Maß für die Genauigkeit eines Modells, welcher zwischen 0 und 1 liegt, wobei 1 die höchste Genauigkeit darstellt
- Angabe, ob es sich bei dem betrachteten Autor um keine Person handelt

# Abgleich

Quelle	TP	FN	FP	TN	Keine Person	F1-Score
Beschreibung	87	5	18	125	15	0,8832
README	81	21	11	143	26	0,8351
CFF	168	13	3	24	3	0,9545
CFF preferred citation	78	8	2	13	0	0,9398
PyPI Maintainer	294	2	32	48	50	0,9453
Python Autoren	155	4	35	43	75	0,8883
Python Maintainer	30	0	5	21	25	0,9231
CRAN Autoren	269	4	3	19	1	0,9872
CRAN Maintainer	193	2	0	0	0	0,9948
BibTeX	4	2	0	0	0	0,8000
Summe	1359	61	109	436	195	0,9411

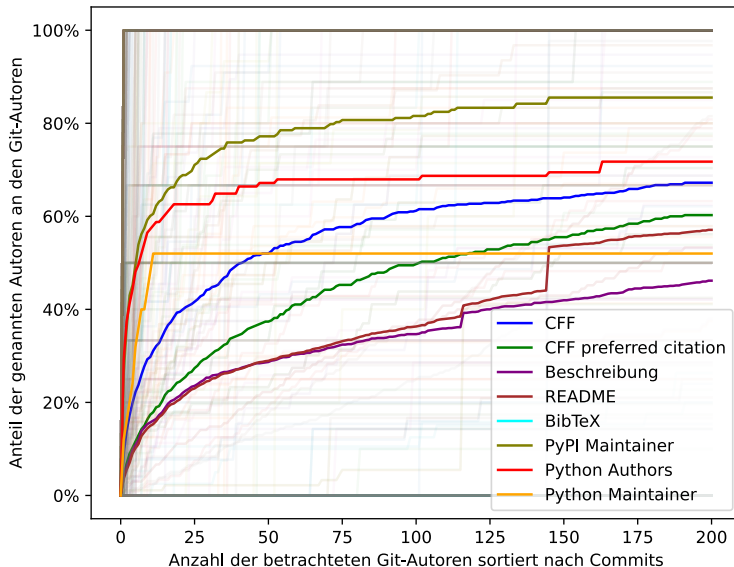
# Top-Git-Autoren an genannten Autoren

- Anteil der Top-Git-Autoren an den genannten Autoren
- Betrachtet PyPI CFF Liste
- Für CFF-Linie gilt: Die Top x Committer sind zu y % in der CFF gelistet



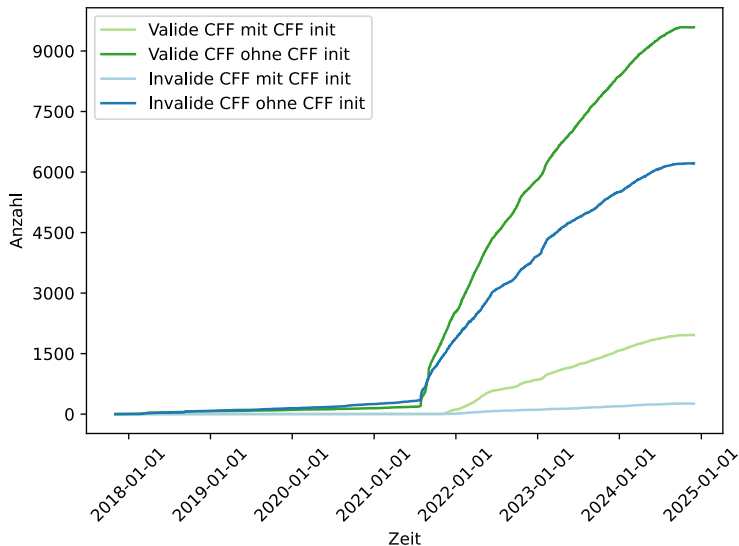
# Genannte Autoren unter Top-Git-Autoren

- Anteil der genannten Autoren unter den Top-Git-Autoren
- Betrachtet PyPI CFF Liste
- Für CFF-Linie gilt: Autoren in der CFF sind zu  $y$  % unter den Top  $x$  Committern



# Validität der CFF-Dateien

- Validität der CFF-Dateien über die Zeit
- Betrachtet alle CFF-Dateien auf GitHub
- Jedes Repository nur einmalig betrachtet



# Diskussion

# Wie gut können Autoren untereinander abgeglichen werden?

- Viele Autoren in Python-Quellen sind keine Personen, sondern Organisationen
- Diese verschlechtern die Ergebnisse, da sie falsch zugeordnet werden können
- README und Beschreibung haben schlechte F1-Scores, was in der NER begründet liegt, da diese bereits Fehler macht
- Viele FP in den Ergebnissen, durch die Verwendung von Python *in* und Autoren mit wenigen Buchstaben in Git
- Im Allgemeinen funktioniert der Abgleich gut, da ein F1-Score von 0,9411 berechnet wurde, welcher größer als 0,9 ist
- F1-Score könnte noch besser sein, ohne die genannten Gegebenheiten



# Was muss ein Softwareentwickler leisten, um als Autor genannt zu werden?

- Frage lässt sich nur allgemeingültig beantworten, da Daten allgemeingültig untersucht wurden
- Autoren mit vielen Commits werden häufiger als Autor genannt → viele Commits haben
- Allerdings: Autoren mit den meisten Commits werden auch manchmal nicht genannt
- Bei der Gründung des Pakets dabei sein, allerdings im Nachhinein nicht umsetzbar
- Autoren werden nachträglich selten hinzugefügt

# Wie gut werden Autoren in den einzelnen Quellen gepflegt?

- In allen Paketen der 5 Listen nur 9 Autoren nachträglich dem CFF oder der BibTEX-Datei hinzugefügt
- Hohe Anzahl der invaliden CFF zeigt, dass die Pflege der Autoren nicht besonders wichtig ist
- Allgemein: Autoren in den betrachteten Listen werden nicht aktiv gepflegt und keine automatischen Prozesse vorhanden
- Begründung: Viele Pakete nicht in der Wissenschaft entstanden, sondern bspw. in Unternehmen

# Fazit

- Diskrepanzen bspw. bei der Nennung der Top-Git-Autoren in den Datenquellen
- Ein Abgleich von Autoren mit wenigen Datenpunkten (Name, E-Mail, Benutzername) ist möglich (**F1**)
- Um als Autor genannt zu werden, ist es gut viele Commits getätigt zu haben, allerdings ist eine Nennung nicht garantiert (**F2**)
- Autoren werden kaum gepflegt (**F3**)
- Verbesserungsbedarf bei der Pflege und Nennung von Autoren in OSS
- Arbeit eines jeden Autors sollte angemessen gewürdigt werden

# Literaturverzeichnis

- [1] Ariel Miculas. *How I got robbed of my first kernel contribution*. 27. Sep. 2023. URL: <https://ariel-miculas.github.io/How-I-got-robbed-of-my-first-kernel-contribution/> (besucht am 03. 06. 2024).
- [2] Arfon M. Smith, Daniel S. Katz und Kyle E. Niemeyer. „Software citation principles“. In: *PeerJ Computer Science* 2 (19. Sep. 2016), e86. DOI: [10.7717/peerj-cs.86](https://doi.org/10.7717/peerj-cs.86).
- [3] Jannik Lindner. *Version Control Systems Industry Statistics*. 3. Mai 2024. URL: <https://worldmetrics.org/version-control-systems-industry-statistics/> (besucht am 21. 05. 2024).
- [4] Jurriaan H. Spaaks u. a. *cffinit*. 8. Aug. 2023. DOI: [10.5281/zenodo.8224012](https://doi.org/10.5281/zenodo.8224012).
- [5] Jurriaan H. Spaaks u. a. *cffconvert*. 22. Sep. 2021. DOI: [10.5281/zenodo.5521767](https://doi.org/10.5281/zenodo.5521767).