TECHNISCHE UNIVERSITÄT
CHEMNITZ

Fakultät für Informatik

CSR-19-07

Analyse verschiedener Distanzmetriken zur Messung des Anonymisierungsgrades θ

Martin Eisoldt · Carsten Neise · Andreas Müller

Juni 2019

Chemnitzer Informatik-Berichte

Analyse verschiedener Distanzmetriken zur Messung des Anonymisierungsgrades θ

Martin Eisoldt
profi.com AG business solutions
meisoldt (at) proficom.de

Dr. Carsten Neise
profi.com AG business solutions
cneise (at) proficom.de

Dr. Andreas Müller
Technische Universität Chemnitz
dr.andreas.mueller (at) informatik.tu-chemnitz.de

***Zusammenfassung*—Das bereits existierende Konzept [4] zur Bewertung der Anonymisierung von Testdaten wird in dieser Arbeit weiter untersucht. Dabei zeigen sich die Vor- und Nachteile gegenüber bereits existierenden Distanzmetriken. Weiterführend wird untersucht, welchen Einfluss Parameteränderungen auf die Ergebnisse haben.**

***Index Terms*—DSGVO, Anonymisierung, Testdaten, Levenshtein**

I. Einleitung

Um realitätsnahe Softwaretests zu ermöglichen, ist es nötig, dass Testdaten mit einer realen Struktur zum Einsatz kommen. Allerdings können aufgrund von datenschutzrechtlichen Einschränkungen keine echten Daten genutzt werden. Vielmehr bedarf es einer Anonymisierung dieser Daten, damit sie unter anderem der aktuellen Datenschutzgrundverordnung (DSGVO) [7] einhalten. Entsprechende Modifizierungen können mit verschiedenen, auf dem Markt verfügbaren, Tools durchgeführt werden[1]. So wurde für diese Arbeit das Tool Tricentis TDM Studio (ehemals Q-Up Studio)[2] verwendet. Dieses kann aber die Verfremdung und somit die Güte der Anonymisierung der Daten nicht bewerten. Von der profi.com AG wurde ein Konzept entwickelt, das es ermöglicht, solche Berechnungen durchzuführen [4]. Dabei wurde eine neue Berechnungsmethode eingeführt, die nicht nur die Ähnlichkeit von Zeichenketten (String) berechnen kann, sondern auch von den bekannten Datentypen Integer, Float, Boolean und Date. Bekannte Distanzmetriken wie die Hamming- [3], Levenshtein- [6] oder Damerau-Levenshtein Distanz [1] hingegen sind nur für Strings geeignet (siehe Tabelle I). Im Folgenden wird die entwickelte Distanzmetrik als Heinrich-Distanz bezeichnet.

Tabelle I
NATIV UNTERSTÜTZTE DATENTYPEN VON DISTANZMETRIKEN

	Hamming	Levenshtein	Damerau	Heinrich
String	✓	✓	✓	✓
Date	X	X	X	✓
Boolean	X	X	X	✓
Integer	X	X	X	✓
Float	X	X	X	✓

[1]https://www.softwaretestinghelp.com/tools/14-test-data-management-tools/

[2]https://documentation.tricentis.com/tdmstudio/1210/en/content/tdm_studio/tdm_about.htm

Der Heinrich-Distanz liegen dabei für die unterschiedlichen Datentypen leicht modifizierte Formeln zu Grunde, welche als Basis eine Gauß-Funktion inne haben und einen anzupassenden Normierungsparameter σ. Diese können im Detail [4] entnommen werden. Die Ausgangsformel für den Anonymisierungsgrad ist dabei:

$$\theta = 1 - e^{-\frac{1}{2}(\frac{d(x)-d(x')}{\sigma})^2} \quad (1)$$

Hierbei ist folgendes definiert: x : Originaldaten, x' : anonymisierte Daten, *d()* : Bewertungsfunktion, σ : Normierungsparameter, $\theta \in [0, 1]$
Ziel ist es, σ so einzustellen, dass θ den optimalen Grad der Anonymisierung bestimmt. Dieser Grad kann empirisch ermittelt werden, indem man σ über verschiedene Durchläufe entsprechend dem Grad der Anonymisierung einstellt, welcher am besten die Realität widerspiegelt.
Aufgrund dieser Anpassung kann für jeden Datentyp jeweils ein anderer Wert für σ genutzt werden. Für Strings wird neben der Länge auch der Inhalt (in Form von Häufigkeit der Zeichen sowie deren Anordnung) betrachtet. In diesem Fall ist es auch möglich, σ noch weiter zu differenzieren. Im Unterschied dazu sind die Repäsentanten der Datentypen Integer und Float zu sehen, bei denen die Differenz zwischen Ausgangswert und verfälschtem Wert entscheidend für die Bewertung ist. Damit bedarf es keiner weiteren Differenzierung von σ [4].
Ziel der vorliegenden Arbeit ist, eine detailliertere Untersuchung der Heinrich-Distanz. Im Fokus stehen dabei der Vergleich mit anderen Distanzmetriken sowie eine Betrachtung des Einflusses von σ auf die Ergebnisse.

II. Vergleich der Distanzmetriken

Die Tests wurden mit einer virtuellen Maschine (VM) auf einem Windows 10-Server durchgeführt. Diese VM hat 8 CPUs und 8 GB RAM. Die Arbeitsweise sah hierbei wie folgt aus. Es wurden aus einem Basisdatensatz bestehend aus 349 Einträgen und 10 Spalten jeweils anonymisierte Daten bewusst verfälscht. Im Nachgang wurde der Originaldatensatz und der verfälschte Datensatz in jeweils eine Datenbank geladen und über das Tool verglichen [4]. Dies wurde für jede der Anonymisierungen durchgeführt, um verschiedene Grade der Anonymisierung zu messen.

Bei den Tests wurden ausschließlich die E-Mail Adressen verglichen, um in einer ersten Form alle Distanzmetriken

miteinander vergleichen zu können. Die Vergleiche wurden mit fünf verschiedenen Datensätzen durchgeführt, wobei die Verfälschung nachfolgend erläutert ist. Ziel ist es, die Anonymisierung suksessive zu erhöhen, um so den Einfluss von σ auf θ messen zu können. Folgende Verfälschungen wurden an den Mail-Adressen durchgeführt: (1) Entfernen des ersten Zeichens, (2) Vertauschen der Zeichen an Position 2 und 3, (3) Entfernen des fünften Zeichens, (4) Vertauschen der Zeichen des Länderkürzels, (5) Entfernen des letzten Zeichens. Für diese erste Bewertung der Heinrich-Distanz im Rahmen des Performanzvergleiches mit den anderen Distanzmetriken wurde der Einfachheit halber σ immer auf 1 gesetzt.

A. Vergleich der Ergebnisse

In einem ersten Schritt wurden die einzelnen Distanzmetriken auf ihre Bewertung der Änderung der verfälschten Datensätzen untersucht. Abbildung 1 zeigt, dass die Hamming-Distanz den verschiedenen Änderungen immer die höchsten Werte zuweist. Dies liegt daran, dass diese Distanzmetrik lediglich betrachtet, an wie vielen Stellen sich zwei Zeichenketten unterscheiden. Um die Ergebnisse der anderen Metriken besser darstellen zu können, wurde in Abbildung 2 die Hamming-Distanz entfernt. Es ist erkennbar, dass die Heinrich-Distanz auf verschiedene Änderungen (vor allem für (2) und (4)) unterschiedlich stark reagiert, während die Ergebnisse für die Levenshtein- und Damerau-Levenshteindistanz keine Änderungen für die verschiedenen Verfälschungen aufzeigen.

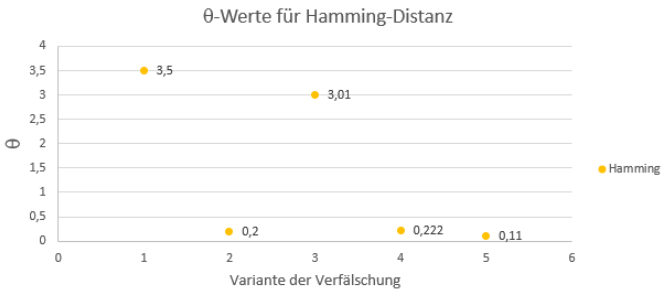

Abbildung 1. Werte mit Hamming

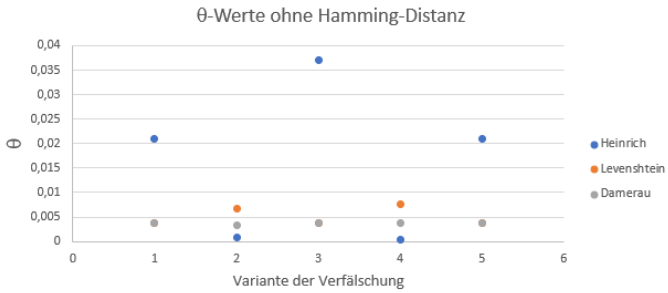

Abbildung 2. Werte ohne Hamming

B. Performance

In Abbildung 3 ist die Auslastung des CPUs für die einzelnen Metriken und Datensätze dargestellt, welche sich für alle vier genutzten Distanzmetriken nur kaum merklich unterscheidet. Die Auslastung des Arbeitsspeichers ist bei allen durchgeführten Tests immer gleich hoch.

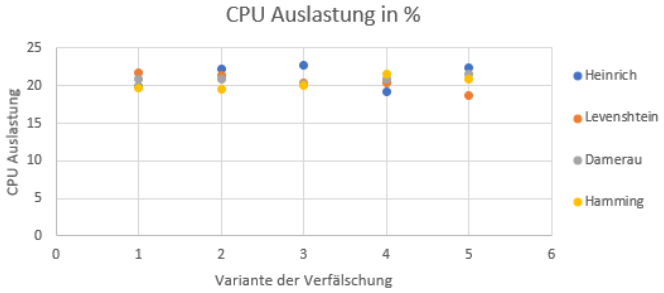

Abbildung 3. CPU Auslastung

Außerdem ist die Dauer der Berechnungen von Interesse. Dabei hat sich gezeigt, dass die Hamming-Distanz die kürzeste Berechnungsdauer von im Mittel 300 ms benötigt hatte. Aufgrund der höheren Komplexität der Berechnung liegt die Heinrich-Distanz hier höher, im Schnitt bei 425 ms, die Berechnungsdauern für die anderen beiden Distanzmetriken liegen dazwischen (siehe Abbildung 4).

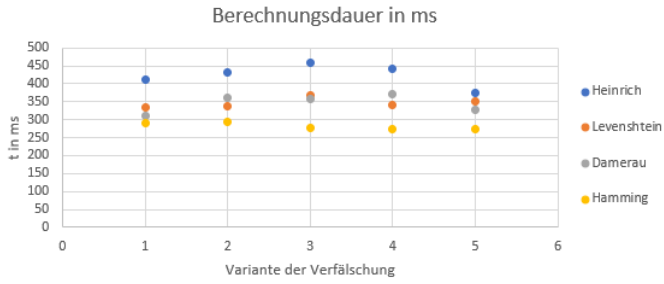

Abbildung 4. Berechnungsdauer

C. Untersuchung von σ

Ziel dieser Untersuchung ist es, einen Eindruck zu gewinnen, wie σ die Ergebnisse des Anonymisierungsgrades beeinflusst. Dafür wurden die Strings auf eine Länge von 10 Zeichen begrenzt, damit eine Vergleichbarkeit für die Messung von θ gegeben ist. Für σ wurden die Werte 0,1; 0,5; 1; 2; 5 und 10 betrachtet. Zusätzlich wurden die Datensätze auf verschiedene Arten verändert. Dies umfasst sowohl das Entfernen von einzelnen Zeichen an bestimmten Positionen als auch das Entfernen von mehreren aufeinander folgenden Zeichen. Des weiteren wurde betrachtet, wie sich die Ergebnisse verhalten, wenn ab einer bestimmten Position die restlichen Zeichen abgeschnitten werden. In einem weiteren Datensatz wurden zeitgleich am Anfang und Ende eines Strings Zeichen entfernt. Anhand der Resultate ist erkennbar, dass es bei einem

großen σ nur geringe Veränderungen für θ gibt, wenn nur Veränderungen an einer oder zwei Positionen vorgenommen werden (siehe Abbildungen 5 und 6). Dies ist unter anderem damit zur erklären, dass die zugrundeliegende Formel 1 eine Gaußfunktion abbildet und mit zunehmenden σ abklingt.
Werden Veränderungen an Positionen vorgenommen, die nicht direkt zusammenhängen (beispielsweise an Beginn und Ende der Zeichenkette), werden die Ergebnisse beim Einsatz eines großen σs genauer abgestuft. Dafür konvergiert θ bei geringeren Werten für σ schneller, wie Abbildung 7 aufweist. Dabei wurde vor beziehungsweise nach folgenden Buchstaben abgeschnitten: (1) 1 & 10; (2) 2 & 9; (3) 3 & 8; (4) 4 & 7; (5) 5 & 6.
In Abbildung 5 ist ein starker Anstieg für θ zu Beginn zu erkennen. Die Ursache dafür ist, dass darauf überprüft wird, ob ein String A in String A' enthalten ist. Beispielsweise wäre dies für A = „Hallo“ und A' = „allo“ der Fall. Eine solche Veränderung wird als schlechte Anonymisierung eingestuft. Gleiches gilt auch, wenn die Zeichenketten ausschließlich am Ende verändert werden.

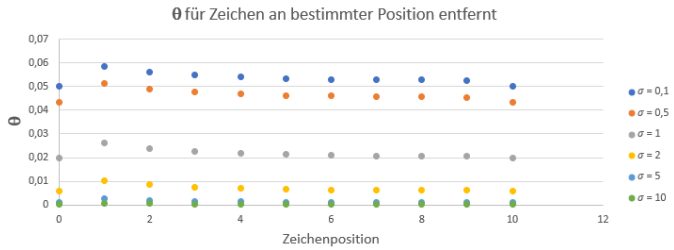

Abbildung 5. Entfernen von Zeichen an bestimmter Position

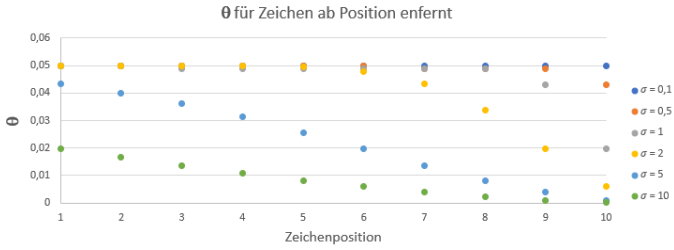

Abbildung 6. Entfernen von Zeichen ab bestimmter Position

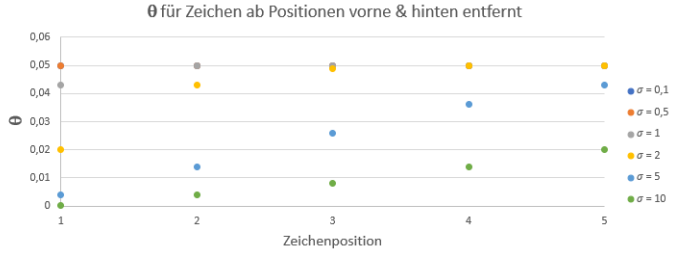

Abbildung 7. Zeichen vorne und hinten ab Positionen entfernt

III. Diskussion

Die in Abschnitt II-C durchgeführten Untersuchungen zeigen, dass verschiedene Werte von σ zu unterschiedlichen Ergebnissen führen können. So existiert die Gefahr, das σ so eingestellt wird, dass eine in der Realität mäßig gute Anonymisierung durch die Heinrichmetrik als sehr gute Anonymisierung bewertet wird. Um solchen Fällen vorzubeugen, ist es empfehlenswert sich an den Ergebnissen der hier behandelten Untersuchung zu orientieren. Somit kann verhindert werden, dass es zu solchen fehlerhaften Bewertungen kommt.
Ein großer Vorteil, der die Nutzung der Heinrich-Distanz gegenüber den anderen Distanzmetriken bietet, ist die native Unterstützung der verschiedenen Datentypen. Damit andere Metriken beispielsweise für Zahlen angewendet werden können, müssen diese zuerst in einen String konvertiert werden. Das entwickelte Tool hingegen erkennt den Datentyp automatisch und führt dann entsprechende Berechnungen durch. Dadurch werden Zeitunterschiede, wie in Abbildung 4 sichtbar, ausgeglichen, da die manuelle Konvertierung für die anderen Metriken zusätzlich Zeit benötigt. Außerdem ermöglicht diese Unterscheidung die Spezifizierung verschiedener σ Werte für die einzelnen Datentypen und somit eine genauere Gewichtung der Berechnungen.

IV. Ausblick

Die bisherige Umsetzung des Tools ist zur Zeit noch nicht praxistauglich. Dies liegt unter anderem daran, dass nur eine vertikale Skalierung möglich ist. Diese stößt aber vergleichsweise schnell an ihre Grenzen. Aus diesem Grund wäre eine horizontale Skalierung wünschenswert. Somit würden verschiedenen Recheneinheiten miteinander kombiniert. Als Architektur eignet sich dafür eine serviceorientierte Architektur. Dabei wird für jeden Datentyp ein eigener Vergleichsservice eingerichtet. Dadurch können je nach Bedarf zusätzliche Serviceeinheiten für den Vergleich hinzugefügt werden.
Die aktuelle Implementierung des Tools basiert auf Java. Somit bedeutet aber, dass für jeden Datensatz, der bewertet werden soll, eine neue Klasse erstellt werden muss. Das bedeutet erhöhten Aufwand für den Nutzer. Es wäre von Vorteil, wenn keine extra Klassen benötigt werden und der Nutzer nur die gewünschten Datensätze in einem ersten Schritt auswählen muss. Dafür müsste das Tool in einer anderen Programmiersprache, zum Beispiel Python, umgesetzt werden.
Für die Berechnungen sollte außerdem berücksichtigt werden, dass für menschliches Lesen besonders der erste und letzte Buchstabe eines Wortes wichtig sind, während die Anordnung der Buchstaben innerhalb eines Wortes weniger relevant ist [5]. Dies wird über eine entsprechende Justierung von σ für den Rand- und Innenbereich eines Wortes erzielt.
Des Weiteren sollte die Bewertung so optimiert werden, dass sie Muster erkennt. Wenn zum Beispiel jedes „a“ aus einem Datensatz entfernt wird, kann dies durchaus zu einem guten

Ergebnis der Anonymisierung einer einzelnen Zeichenkette führen. Allerdings lässt sich aus der Gesamtmenge der Daten ein Muster erkennen und somit die Anonymisierung rückgängig machen. Eine solche Bewertung benötigt die Erweiterung des Algorithmus in dem Umfang, dass dieser auch alle Daten betrachtet.

Außerdem ist anzumerken, dass die Bewertung von einer Booleananonymisierung aktuell nicht als solche zu betrachten ist. So wird von einer perfekten Anonymisierung ausgegangen, wenn ein Wert auf den anderen gesetzt wird. Dies lässt aber noch immer zu, dass Ergebnisse problemlos auf ihren Ausgangswert zurückgeführt werden können. Vielmehr eignet sich hier der Ansatz der „Differential Privacy“ [2]. Dabei wird beispielsweise bei einem bestimmten Anteil von „false“ zufällig entschieden, ob der Wert verändert wird oder nicht. Somit kann bei keiner Angabe mehr eindeutig festgestellt werden, welcher Wert im Ausgangsdatensatz stand.

LITERATUR

[1] Damerau, Fred J. A technique for computer detection and correction of spelling errors. In: Communications of the ACM. Band 7, Nr. 3, März 1964, S. 171–176

[2] Dwork, Cynthia. Differential privacy. Encyclopedia of Cryptography and Security (2011): 338-340

[3] Hamming, Richard W. Error-detecting and error-correcting codes. In: Bell System Technical Journal, XXIX (2), 1950, S. 147–160.

[4] Heinrich, M. Sc Jan-Philipp. Ähnlichkeitsmessung von ausgewählten Datentypen in Datenbanksystemen zur Berechnung des Grades der Anonymisierung. (2018).

[5] Kinoshita, Sachiko; Lupker, Stephen J. Masked Priming: The State of the Art Macquarie Monographs in Cognitive Science, 2004, S. 53 ff.

[6] Levenshtein, Vladimir I. Binary codes capable of correcting deletions, insertions, and reversals. In: Doklady Akademii Nauk SSSR. Band 163, Nr. 4, 1965, S. 845–848 (Russisch, Englische Übersetzung in: Soviet Physics Doklady, 10(8) S. 707–710, 1966).

[7] Parlament, Europäisches, Verordnung (EU) 2016/679 des Europäischen Parlaments und des Rates vom 27. April 2016 zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten, zum freien Datenverkehr und zur Aufhebung der Richtlinie 95/46/EG (Datenschutz-Grundverordnung), 2016

Chemnitzer Informatik-Berichte

In der Reihe der Chemnitzer Informatik-Berichte sind folgende Berichte erschienen:

CSR-13-01 Navchaa Tserendorj, Uranchimeg Tudevdagva, Ariane Heller, Grenzgänger - Integration of Learning Management System into University-level Teaching and Learning, Januar 2013, Chemnitz

CSR-13-02 Thomas Reichel, Gudula Rünger, Multi-Criteria Decision Support for Manufacturing Process Chains, März 2013, Chemnitz

CSR-13-03 Haibin Xu,Thomas Reichel, Gudula Rünger, Michael Schwind, Softwaretechnische Verknüpfung der interaktiven Softwareplattform Energy Navigator und der Virtual Reality Control Platform, Juli 2013, Chemnitz

CSR-13-04 International Summerworkshop Computer Science 2013, Proceedings of International Summerworkshop 17.7. - 19.7.2013, Juli 2013, Chemnitz

CSR-13-05 Jens Lang, Gudula Rünger, Paul Stöcker, Dynamische Simulationskopplung von Simulink-Modellen durch einen Functional-Mock-up-Interface- Exportfilter, August 2013, Chemnitz

CSR-14-01 International Summerschool Computer Science 2014, Proceedings of Summerschool 7.7.-13.7.2014, Juni 2014, Chemnitz

CSR-15-01 Arne Berger, Maximilian Eibl, Stephan Heinich, Robert Herms, Stefan Kahl, Jens Kürsten, Albrecht Kurze, Robert Manthey, Markus Rickert, Marc Ritter, ValidAX - Validierung der Frameworks AMOPA und XTRIEVAL, Januar 2015, Chemnitz

CSR-15-02 Maximilian Speicher, What is Usability? A Characterization based on ISO 9241-11 and ISO/IEC 25010, Januar 2015, Chemnitz

CSR-16-01 Maxim Bakaev, Martin Gaedke, Sebastian Heil, Kansei Engineering Experimental Research with University Websites, April 2016, Chemnitz

Chemnitzer Informatik-Berichte

CSR-18-01 Jan-Philipp Heinrich, Carsten Neise, Andreas Müller, Ähnlichkeitsmessung von ausgewählten Datentypen in Datenbanksystemen zur Berechnung des Grades der Anonymisierung, Februar 2018, Chemnitz

CSR-18-02 Liang Zhang, Guido Brunnett, Efficient Dynamic Alignment of Motions, Februar 2018, Chemnitz

CSR-18-03 Guido Brunnett, Maximilian Eibl, Fred Hamker, Peter Ohler, Peter Protzel, StayCentered - Methodenbasis eines Assistenzsystems für Centerlotsen (MACeLot) Schlussbericht, November 2018, Chemnitz

CSR-19-01 Johannes Dörfelt, Wolfram Hardt, Christian Rosjat, Intelligente Gebäudeklimatisierung auf Basis eines Sensornetzwerks und künstlicher Intelligenz, Februar 2019, Chemnitz

CSR-19-02 Martin Springwald, Wolfram Hardt, Entwicklung einer RAD-Plattform im Kontext verteilter Systeme, März 2019, Chemnitz

CSR-19-03 André Böhle, René Schmidt, Wolfram Hardt, Evaluation von Signaleigenschaften zur Lokalisierung von Einschlägen mit Piezokeramischen Sensoren, März 2019, Chemnitz

CSR-19-04 Johannes Götze, René Schmidt, Wolfram Hardt, Hardwarebeschleunigung von Matrixberechnungen auf Basis von GPU Verarbeitung, März 2019, Chemnitz

CSR-19-05 Vincent Kühn, Reda Harradi, Wolfram Hardt, Expert System for Adaptive Flight Missions, Juni 2019, Chemnitz

CSR-19-06 Samer Salamah, Guido Brunnett, Christian Mitschke, Tobias Heß, Synthesszing gait motions from spline-based progression functions of controlled shape, Juni 2019, Chemnitz

CSR-19-07 Martin Eisoldt, Carsten Neise, Andreas Müller, Analyse verschiedener Distanzmetriken zur Messung des Anonymisierungsgrades θ, Juni 2019, Chemnitz

Chemnitzer Informatik-Berichte

ISSN 0947-5125

Herausgeber: Fakultät für Informatik, TU Chemnitz
Straße der Nationen 62, D-09111 Chemnitz