

Analyse und Visualisierung der OCR-Konfidenz für Dokumente aus den Digitalen Beständen der Staats- und Universitätsbibliothek Hamburg

Abschlussarbeit im Qualifikationskurs „Data Librarian“ an der TH Köln

Abstract

Diese Abschlussarbeit beschreibt die notwendigen Schritte um die OCR-Konfidenz eines retrodigitalisierten Werkes zu analysieren und zu visualisieren. Es werden der Dateidownload, die Datenextraktion, die statistische Auswertung und Visualisierung sowie die Generierung eines Reports übersichtsartig dargestellt. Das so beschriebene Verfahren eignet sich als Indikator, erfordert aber stets die intellektuelle Kontrolle. Bibliotheken könnten damit jedoch gezielter Fehlerquellen ausfindig machen, beheben und somit langfristig die OCR-Qualität verbessern.

Veröffentlichungshinweise

Die hierbei entstandene Software OCA.py (**O**CR **C**onfidence **A**nalysis script written in **python**) wird auf <https://github.com/michaelkubina/ocapy> im `graduation_work`-Branch unter GPL3 (<https://www.gnu.org/licenses/gpl-3.0.en.html>) veröffentlicht, zur Abschlussarbeit gehörende und sonstige Dokumente unter CC-BY-SA 4.0 (<https://creativecommons.org/licenses/by/4.0/>).

Inhaltsverzeichnis

Abstract	1
Veröffentlichungshinweise.....	1
Zielsetzung.....	2
Vorgehen	2
Schritt 1 – Dateidownload	2
Schritt 2 – Extraktion der Wortkonfidenz	3
Schritt 3 – Statistik.....	3
Schritt 4 – Visualisierung	4
Schritt 5 – Report.....	5
Schritt 6 – Anwendungstest	7
Literaturverzeichnis	8

Zielsetzung

Das Ziel dieser Abschlussarbeit ist es am Beispiel von retrodigitalisierten Dokumenten der Staats- und Universitätsbibliothek Hamburg die vermutete Qualität von OCR-Ergebnissen statistisch auszuwerten und die Ergebnisse zusammengefasst in einem Report zu visualisieren. Dies ist möglich für Dokumente, bei denen die OCR-Engine Konfidenzwerte ausgegeben hat. Die OCR-Konfidenz bemisst den Grad der Sicherheit einer OCR-Engine die jeweiligen Textkomponenten richtig erkannt zu haben (IMPACT, 2022). Die OCR-Konfidenz gibt somit nicht zwangsläufig Auskunft darüber, wie hoch die tatsächlichen Erkennungsraten waren und ob die Volltexte korrekt sind, sondern ist höchstens ein Indikator dafür. Dennoch kann die Erkennungssicherheit wichtige Rückschlüsse dafür liefern, wo potentiell schlechte OCR-Ergebnisse vorliegen könnten.

Vorgehen

Für die Analyse und Visualisierung der Ergebnisse wird ein mehrstufiges Python-Script geschrieben, welches auf Bibliotheken für Dateihandling, Datenanalyse und Datenvisualisierung zurückgreift.

Schritt 1 – Dateidownload

Im ersten Schritt muss die METS-Datei für ein Digitalisat heruntergeladen werden, die sich aus dem METS-Schema ableitet (LIBRARY OF CONGRESS, 2022a) und die durch einen eindeutigen Identifier adressierbar ist – den sogenannten *Record Identifier*. In der von der DFG vorgeschriebenen METS-Datei werden strukturelle und deskriptive Metadaten erfasst, durch die das digitale Objekt vollumfänglich beschrieben und zusammengehalten wird. Deskriptive Metadaten liegen dabei in der METS-Datei im MODS-Format (LIBRARY OF CONGRESS, 2022b) vor, so dass auch von einer METS/MODS-Datei gesprochen wird (DEUTSCHE FORSCHUNGSGEMEINSCHAFT, 2016, S. 30 ff.). Über diese METS/MODS-Datei können somit aus den strukturellen Metadaten die URLs für die OCR-Volltexte extrahiert und anschließend auch diese heruntergeladen werden. Die OCR-Volltexte liegen hierbei im Normalfall im ALTO-Format vor, beschrieben durch das ALTO-Schema (LIBRARY OF CONGRESS, 2022c). Im Script wird daher per *requests* (REITZ, 2022) zunächst die METS-Datei heruntergeladen (sofern nicht bereits eine lokale Kopie vorliegt) und via *os* gespeichert. Mittels *beautifulsoup* (RICHARDSON, 2022) werden die Volltext-URLs extrahiert und alle ALTOs heruntergeladen.

```
Retrieving: https://img.sub.uni-hamburg.de/kitodo/PPN1026788544/00000001.xml -> Done!  
Retrieving: https://img.sub.uni-hamburg.de/kitodo/PPN1026788544/00000002.xml -> Done!  
Retrieving: https://img.sub.uni-hamburg.de/kitodo/PPN1026788544/00000003.xml -> Done!  
Retrieving: https://img.sub.uni-hamburg.de/kitodo/PPN1026788544/00000004.xml -> Done!  
Retrieving: https://img.sub.uni-hamburg.de/kitodo/PPN1026788544/00000005.xml -> Done!  
Retrieving: https://img.sub.uni-hamburg.de/kitodo/PPN1026788544/00000006.xml -> Done!
```

Abbildung 1: Beispiel eines Dateidownloads der Volltextseiten (Quelle: eigene Darstellung)

Schritt 2 – Extraktion der Wortkonfidenz

Im zweiten Schritt werden mittels `beautifulsoup` alle Wortkonfidenzen aus den ALTO-Dateien extrahiert. Diese werden (falls vorhanden) im `WC`-Attribut (= Word Confidence) des `String`-Elements (einzelnes Wort) als eine Fließkommazahl zwischen 0.0 (unsicher) und 1.0 (sicher) gespeichert (LIBRARY OF CONGRESS, 2022c). Wir extrahieren aus den heruntergeladenen Volltexten bzw. den lokalen Kopien für jedes Wort jeder Zeile (`TextLine`-Element) einer jeden Seite eines Dokuments die Wortkonfidenzen und legen diese in einer verschachtelten Liste ab. Auf oberster Ebene entspricht jeder Listeneintrag einer Volltextseite. Darunter liegt eine weitere Liste für jede Textzeile dieser Seite. Jede Textzeile hat wiederum eine neue Liste mit allen Wortkonfidenzen der Wörter dieser Zeile. Diese verschachtelte Liste wird rekursiv mittels verschachtelter Schleifen erzeugt. Alle Einträge sind bis hierhin noch als Datentyp „String“ hinterlegt.

Nun wird mit `pandas` aus den extrahierten Wortkonfidenzen eine Liste aus DataFrames erzeugt, bei der jeder DataFrame einer Seite mit allen Textzeilen und deren Wortkonfidenzen entspricht. Diese DataFrames können sodann bereinigt, transformiert und statistisch ausgewertet werden. Hierbei ändern wir daher zunächst zur besseren Lesbarkeit die Index-Bezeichnungen für die Zeilen zu „Textline“ und die Spalten zu „Word“, ändern alle leeren Strings mittels `numpy` von „None“ zu „NaN“ = Not a Number (NUMPY DEVELOPERS, 2022), alle Wortkonfidenzen, die als „1.“ herausgeschrieben wurden, zu „1.0“ und im letzten Schritt den elementaren Datentyp von „String“ zu „Float“. Hierdurch ist es nun möglich eine statistische Auswertung vorzunehmen, weil alle Werte als Fließkommazahlen bzw. als für den Datentyp vorgesehenen NULL-Wert vorliegen.

	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7	Word 8	Word 9	Word 10	Word 11
Textline 1	0.37000	0.52400	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Textline 2	0.64000	0.35500	0.38286	0.69000	0.14000	0.25500	0.42000	0.90600	NaN	NaN	NaN
Textline 3	1.00000	0.69200	0.60500	0.68300	0.49600	0.79222	0.64667	0.62714	0.74000	0.47333	NaN
Textline 4	0.76000	0.79000	0.57429	0.63000	0.69800	0.74923	0.74125	0.57333	NaN	NaN	NaN
Textline 5	0.74000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Textline 6	0.59667	0.70636	0.68250	0.51333	0.50500	0.55909	NaN	NaN	NaN	NaN	NaN
Textline 7	0.56333	0.66250	0.60750	0.62500	0.72333	0.82000	NaN	NaN	NaN	NaN	NaN

Abbildung 2: Beispiel eines DataFrames für eine Textseite (Quelle: eigene Darstellung)

Schritt 3 – Statistik

Im dritten Schritt wird eine rudimentäre statistische Auswertung vorgenommen. Zunächst wird ermittelt, wie viele Textzeilen jede Seite hat, und das Ergebnis in einer Liste gespeichert.

Anschließend wird für jeden DataFrame, sprich für jede Seite, mittels `describe()` eine einfache statistische Beschreibung erzeugt. Wir erhalten hierbei den Durchschnitt, die Standardabweichung,

die Quantilen (inkl. Median = 50%) und die Anzahl der Wörter pro Seite. Damit dies möglich wird, müssen pro DataFrame alle Wortkonfidenzen per `stack()` in eine einzige Spalte zusammengefasst und per `dropna()` von NULL-Werten bereinigt werden. Zum Schluss wird die Anzahl an Textzeilen an den resultierenden DataFrame für die Statistik angehängt und der Index für die Volltextseiten entsprechend umbenannt.

	count	mean	std	min	25%	50%	75%	max	textlines
Page 1	4.0	0.78099	0.10183	0.68714	0.72379	0.75675	0.81396	0.92333	1
Page 2	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0
Page 3	21.0	0.43762	0.18741	0.00000	0.32250	0.44000	0.54200	0.92000	7
Page 4	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0
Page 5	58.0	0.57684	0.19114	0.13750	0.44857	0.57812	0.71000	1.00000	18
Page 6	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0
Page 7	28.0	0.69808	0.18984	0.27750	0.59063	0.75375	0.80313	1.00000	8

Abbildung 3: Beispiel für die allgemeine Seitenstatistik (Quelle: eigene Darstellung)

Schritt 4 – Visualisierung

Im vierten Schritt werden die Werte in Anlehnung an die Klimastreifen von Ed Hawkings visualisiert, wo die globale Temperaturentwicklung in Form von aneinandergereihten vertikalen Streifen dargestellt wird. Die Farbe eines jeden Streifen hängt davon ab, inwieweit dieser von einem gegebenen Mittelwert abweicht (HAWKINS, 2018). In diesem Projekt wird daher das Gesamtdokument so visualisiert, dass jeder Streifen einer Textseite entspricht und die Abweichung der durchschnittlichen OCR-Konfidenz der gesamten Seite farblich hervorgehoben wird. Hierfür wird mit `matplotlib` eine Grafik erzeugt und in einem von uns angelegten Bildverzeichnis abgelegt. Für die Grafik wird auf eine Lösung zurückgegriffen, die im Matplotlibblog veröffentlicht wurde (NÖTHE, 2019). Es werden jedoch Anpassungen vorgenommen, da wir keine historischen Klimadaten haben, zu denen wir die Werte in Relation setzen und klar definierte Limits haben. Wir verwenden zudem eine andere `colormap`, die für diesen Zweck sinnvoller scheint. Die Farbskala geht linear von Blau (0.0) über Rot (0.5) zu Grün (1.0), so dass bereits eine Abweichung in den ersten beiden Quantilen klar als solche zu erkennen ist.



Abbildung 4: Beispielvisualisierung für die mittlere OCR-Konfidenz der Seiten eines Buche (Quelle: eigene Darstellung)

Zudem wird für jede Textzeile einer Textseite ebenfalls eine solche Visualisierung vorgenommen, bei der ein Streifen der OCR-Konfidenz eines Wortes dieser Textzeile entspricht. Die so visualisierten Textzeilen werden anschließend zusammengefügt um eine Heatmap für die jeweilige Seite zu erzeugen. Hierfür wird auf eine Lösung mittels `pillow` zurückgegriffen (NKKM, 2022), mittels derer zwei Bilder zu einem Bild zusammengefügt werden können. Mit jedem neuen „Klimastreifen“ wird dieser an das vorherige (bereits zusammengefügte) Bild angehängt, bis am Ende die Heatmap für die gesamte Seite fertig ist und das Ergebnis per `shutil` in unser Ausgabeverzeichnis kopiert wird. Man muss dazu anmerken, dass dieses Verfahren optimierungsbedürftig ist, weil viele Schreib-/Leseoperationen durchgeführt werden müssen, um eine einzige Heatmap zu erzeugen. Im Anschluss erhält man abhängig von der Anzahl der Textzeilen pro Seite unterschiedlich hohe Bilder. Diese werden abschließend nochmal auf eine einheitliche Größe für den Report gebracht.

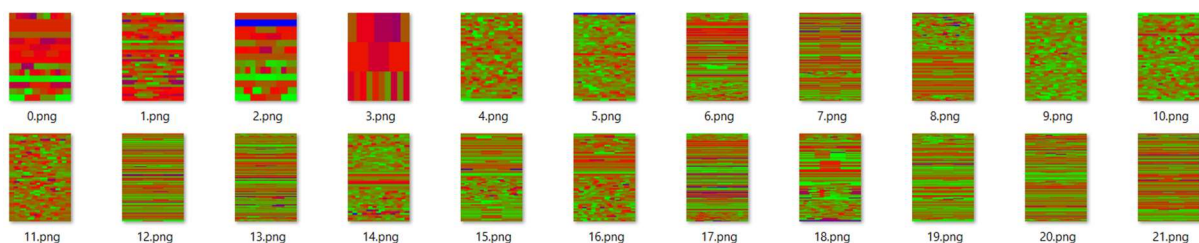


Abbildung 5: Überblick über die Heatmaps in Einheitsgröße (Quelle: eigene Darstellung)

Schließlich nutzen wir unsere Daten um mittels `seaborn` eine Häufigkeitsverteilung der OCR-Konfidenz zu erzeugen (WASKOM, 2022). Hierfür müssen wir nur per `concat()` und `stack()` alle DataFrames in eine große Spalte überführen und können anschließend die Grafik erzeugen.

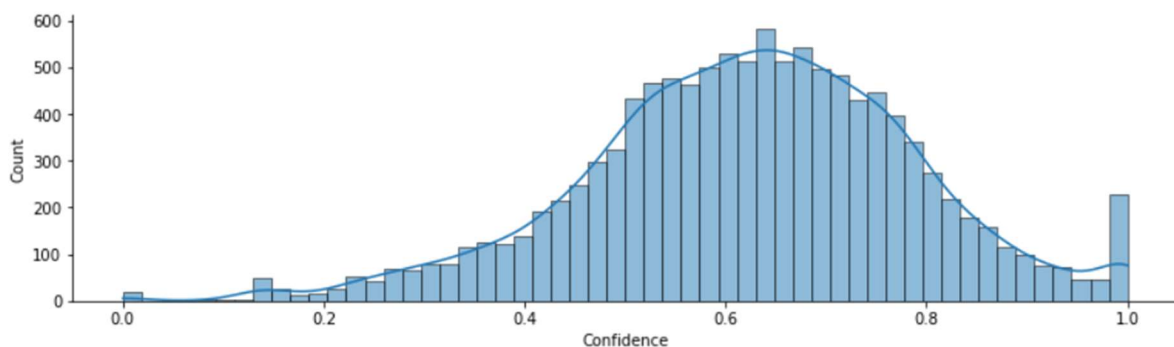


Abbildung 6: Beispiel für eine Häufigkeitsverteilung der OCR-Konfidenz (Quelle: eigene Darstellung)

Schritt 5 – Report

Im fünften Schritt wird ein Report erstellt, der die Ergebnisse der Auswertung auf einer übersichtlichen Seite aggregiert. Hierfür wird ein HTML-Dokument erzeugt und es werden alle generierten Bilder, deskriptive Metadaten und statistischen Daten eingebunden und sinnvoll verlinkt. Der Report nutzt hierbei das CSS-Framework `Bootstrap`, welches bereits fertige CSS-Klassen für

ein attraktives Erscheinungsbild bietet, als auch ein Gridsystem und responsives Design (BOOTSTRAP TEAM, 2022). Wir legen einen Übersichtsblock mit rudimentären Titeldaten (Autor*in, Jahr, Titel) an, sowie Verlinkungen auf die METS-Datei und das Objekt in den Digitalisierten Beständen der Staats- und Universitätsbibliothek Hamburg. Dort sind auch die Häufigkeitsverteilung, die generelle Statistik und der „Klimastreifen“ für das gesamte Buch zu finden (mittlere Konfidenz pro Seite). Darunter liegen alle Einzelseiten als einzelne Kärtchen, mit Seitenstatistiken und den Heatmaps bzw. Leerseiten. Zugleich erfolgt hier eine Verlinkung auf das Originalbild und die darüber erzeugte ALTO-Datei, damit man einen Einblick in die zugrundeliegenden Ausgangsdaten erhält. Damit das HTML-Dokument sauber formatiert ist, wird dieses mittels `prettify()` aus `pprint` optimiert.

Result for PPN1026788544

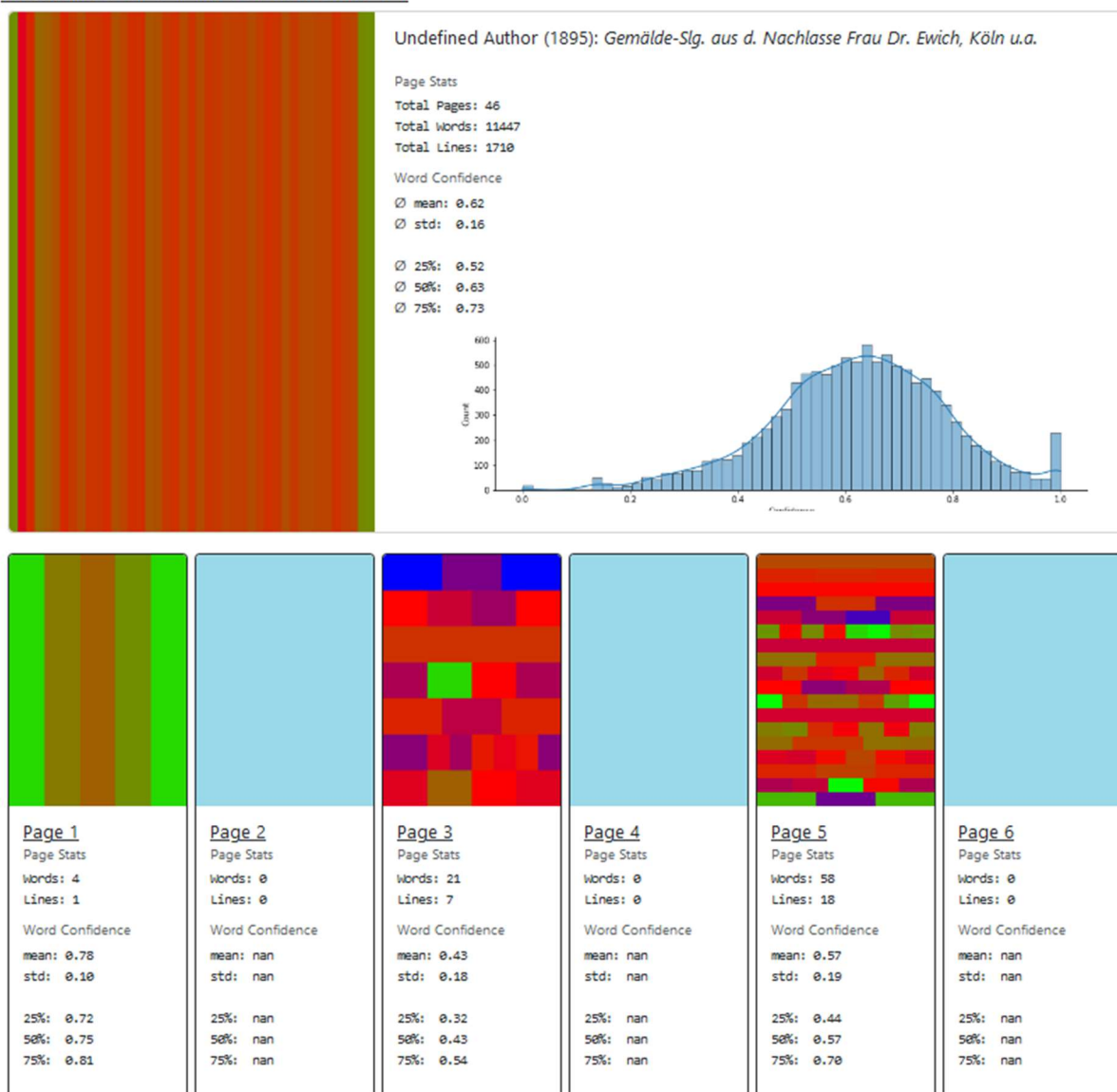



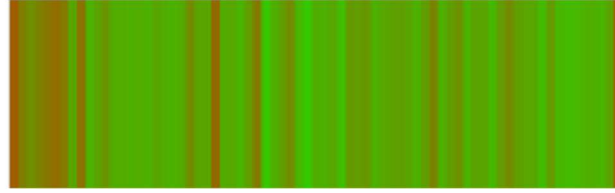


Abbildung 7: Beispiel für einen fertigen Report (Quelle: eigene Darstellung)

Schritt 6 – Anwendungstest

Das Python-Script wurde auf vier ausgewählte Dokumente für eine Ergebnisdiskussion angewandt:

(1895): Gemälde-Slg. aus d. Nachlasse Frau Dr. Ewich	PPN1026788544
 <p>Abbildung 8: Gute OCR, geringe Konfidenz, homogenes Erscheinungsbild</p>	<p>Wir haben trotz einer guten OCR eine geringe Konfidenz. Im Mittel liegt diese bei 62% mit einer absoluten Streuung von +/-16%. Es zeigt sich ein relativ homogenes Bild ohne weitere Auffälligkeiten. Es ist nicht ohne weiteres ersichtlich, weshalb die Konfidenz so gering ausfiel.</p>
Stechow, Wolfgang (1932): Apollo und Daphne	PPN86268370X
 <p>Abbildung 9: Gute OCR, hohe Konfidenz, Anomalie im letzten Drittel</p>	<p>Wir haben eine gute OCR mit hoher Konfidenz. Das erste Quantil liegt bei 71%, der Median bei 79%, das letzte Quantil bei 88%. Im letzten Buchdrittel nimmt die Konfidenz ab. Abbildungen mit Bildunterschriften verzerren die Statistik aufgrund geringer Konfidenz und geringer Wortanzahl.</p>
Deecke, Ernst (1890): Lübsche Geschichten und Sagen	PPN1041860838
 <p>Abbildung 10: Schlechte OCR, geringe Konfidenz, homogenes Erscheinungsbild</p>	<p>Wir haben eine schlechte OCR bei geringer Konfidenz. Der Median fällt mit 49% sehr gering aus und liegt sogar noch unter dem Mittelwert von 50%. Die Standardabweichung beträgt +/-17%. Das erste und letzte Quantil fallen mit 37% bzw. 61% sehr gering aus. Eine Prüfung ergibt, dass ein verkehrtes, wenn auch ähnliches, Schriftsystem verwendet wurde (Lateinisch statt Fraktur).</p>
Mondon-Vidailhet, Casimir (1904): Texte abyssin	PPN1672846668
 <p>Abbildung 11: Schlechte OCR, extrem hohe Konfidenz, statistische und optische Anomalien</p>	<p>Wir haben eine schlechte OCR mit extrem hoher Konfidenz. Der Median liegt bei 97%, das letzte Quantil bei 99%. Der Mittelwert beträgt 81% und die Standardabweichung +/-28%. Die statistische Auffälligkeit wird auch in der Visualisierung der Einzelseiten deutlich, wo sich eine hohe Wortanzahl mit extremer Sicherheit mit einer geringen Wortanzahl mit extremer Unsicherheit abwechseln. Hier wurde ein komplett verkehrtes Schriftsystem benutzt (Lateinisch oder Fraktur statt Ge'ez).</p>

Literaturverzeichnis

BOOTSTRAP TEAM (2022) *Get started with Bootstrap* [online].

<https://getbootstrap.com/docs/5.2/getting-started/introduction/> [Zugriff am: 10.08.2022]

DEUTSCHE FORSCHUNGSGEMEINSCHAFT (2016) *Praxisregel „Digitalisierung“* [online]. Bonn:

Deutsche Forschungsgemeinschaft e.V. https://www.dfg.de/formulare/12_151/12_151_de.pdf

[Zugriff am: 12.08.2022]

HAWKINS, ED (2018) *Warming stripes* [online.]. *Ed Hawkins: Climate Lab Book*. <https://www.climate-lab-book.ac.uk/2018/warming-stripes/>

[Zugriff am: 15.07.2022]

IMPACT Centre of Competence in Digitisation (2022) *Confidence Level (OCR)* [online]. San Vicente del Raspeig: Fundación General de la Universidad de Alicante.

<https://www.digitisation.eu/glossary/confidence-level-ocr/> [Zugriff am: 15.07.2022]

LIBRARY OF CONGRESS (2022a) *METS: Metadata Encoding & Transmission Standard* [online].

Washington, DC: Library of Congress. <http://www.loc.gov/standards/mets/> [Zugriff am: 10.08.2022]

LIBRARY OF CONGRESS (2022b) *MODS: Metadata Object Description Schema* [online]. Washington,

DC: Library of Congress. <https://www.loc.gov/standards/mods/> [Zugriff am: 10.08.2022]

LIBRARY OF CONGRESS (2022c) *ALTO: Technical Metadata for Layout and Text Objects* [online].

Washington, DC: Library of Congress. <https://www.loc.gov/standards/alto/> [Zugriff am: 10.08.2022]

NKMK (2022) *Concatenate images with Python, Pillow* [online]. <https://note.nkmk.me/en/python-pillow-concat-images/>

[Zugriff am: 12.08.2022]

NÖTHE, MAXIMILIAN (2019) *Creating the Warming Stripes in Matplotlib* [online]. Maximilian Nöthe:

matplotlibblog. <https://matplotlib.org/matplotlibblog/posts/warming-stripes/> [Zugriff am: 13.08.2022]

NUMPY DEVELOPERS (2022) *NumPy Documentation* [online]. <https://numpy.org/doc/stable/> [Zugriff

am: 11.08.2022]

REITZ, KENNETH (2022) *Requests: http for Humans™* [online].

<https://requests.readthedocs.io/en/latest/> [Zugriff am: 12.08.2022]

RICHARDSON, LEONARD (2022) *Beautiful Soup* [online]. [https://beautiful-soup-](https://beautiful-soup-4.readthedocs.io/en/latest/)

[4.readthedocs.io/en/latest/](https://beautiful-soup-4.readthedocs.io/en/latest/) [Zugriff am: 13.08.2022]

WASKOM, MICHAEL (2022) *seaborn: statistical data visualization* [online].

<https://seaborn.pydata.org/> [Zugriff am: 10.08.2022]