Verfasser: Florian Landes
Willingweg 8
82436 Eglfing
E-Post: florian.landes@gmx.de
Matrikelnummer: 10585095

Betreuer: Dr. Maximilian Hadersbeck
Centrum für Informations- und Sprachverarbeitung der
Ludwig-Maximilians-Universität München
Oettingenstraße 67
80538 München
E-Post: maximilian@cis.uni-muenchen.de

Bachelorarbeit

für den Studiengang Informatik an der

LMU München

Thema:

Optical Character Recognition (OCR) – Optische Zeichenerkennung (OZE)

Ein Werkzeug zur Verknüpfung von digitaler Edition und Faksimile?

Semiautomatische Ermittlung von Bildkoordinaten für WiTTFind

Zusammenfassung

Die vorliegende Arbeit befasst sich mit der semiautomatischen Ermittlung von Bildkoordinaten für die Suchmaschine WiTTFind mit Hilfe von OCR-Technologie. Zunächst kommen die digitale Wittgenstein-Edition und das Thema Optical Character Recognition zur Vorstellung. Daran anschließend wird die entwickelte Arbeitskette von der Vorbearbeitung der Rohbilder bis zur Nachkorrektur der berechneten Bildkoordinaten nachgezeichnet. Die Erkennung von Typo- und Manuskripten mit OCR- bzw. HTR-Anwendungen war Gegenstand der Betrachtung. Mit einem Shell-Skript lassen sich die erarbeiteten Werkzeuge unter Debian-basierten Systemen installieren. Die errechneten Daten werden sowohl für den Faksimilereader als auch zur Darstellung von Suchergebnissen in WiTTFind verwendet.

Inhaltsverzeichnis

1. Retrodigitalisierung – Eine zentrale Aufgabe der Geisteswissenschaften

Die Digitalisierung ist eine der größten Herausforderungen unserer Zeit. Kein Bereich kann sich der technischen Revolution unserer Tage entziehen. Das gilt selbstverständlich auch für die Geisteswissenschaften. In ihren verschiedenen Disziplinen sind Methoden, Ressourcen und Werkzeuge in ein neues Zeitalter zu überführen und mit den Möglichkeiten der elektronischen Datenverarbeitung zu erweitern. Um neue Erkenntnisse zu gewinnen gilt es, Bewährtes anzupassen und neue Auswertungsmöglichkeiten zu schaffen.

Mittlerweile hat sich der Terminus „Digital Humanities“[1] - digitale Geisteswissenschaften herausgebildet. Er fasst disziplinübergreifend Erscheinungen, die mit dem Phänomen in Verbindung stehen, zusammen und beschreibt gleichzeitig einen neuen Fachbereich.

Eine zentrale Aufgabe für die gesamte akademische Welt ist die Retrodigitalisierung der riesigen Menge an analogen Informationen in Archiven und Bibliotheken. Dabei genügt es nicht, Dokumente im Bild festzuhalten. Es ist vielmehr erforderlich, die in Druckbuchstaben und Handschrift gefassten Texte zu speichern, um so deren Aussagen und Inhalte rechnergestützt verarbeiten zu können.

Anwendungen für diesen Zweck werden meist unter dem Kürzel OCR (Optical Character Recognition) zusammengefasst.[2] Ins Deutsche übersetzt spräche man wohl von Optischer Zeichenerkennung (OZE). Praktisch verstanden wird darunter freilich nicht nur der Vorgang, einzelne Zeichen zu erkennen. Im Mittelpunkt steht die geschickte Kombination beliebig komplexer Algorithmen und meist linguistischer Hilfsmittel zur korrekten Erkennung von Zeichen, Wörtern, Zeilen oder ganzen Seiten.

Dem Sammelbegriff OCR ist eine gewisse Unschärfe zu eigen. Wie umrissen, gehen die verwendeten Anwendungen weit über die eigentliche Zeichenerkennung hinaus. Unberechtigterweise führt der weitaus passendere Begriff Texterkennung im Sprachgebrauch eine untergeordnete Rolle.

Dabei haben sich eigene Teilbereiche zum Beispiel zur Handschrifterkennung herausgebildet. Während die Bearbeitung gedruckter Texte als weitgehend gelöstes Problem angesehen werden kann,[3] ist die zuverlässige Handschrifterkennung mit großen Schwierigkeiten verbunden und bildet einen Schwerpunkt der Forschung.

Bei bisherigen Digitalisierungsprojekten wurden von wenigen Ausnahmen abgesehen gedruckte Werke in elektronische Form überführt. Google Books als weltweit größte Unternehmung zur

1 Digital Humanities im deutschsprachigen Raum e.V., Netzauftritt: https://dig-hum.de/, Stand: 31.07.2019.

2 Vgl. Hierzu: [o. A.], OCR in: Peter Fischer (Hg.), Peter Hofer (Hg.), Lexikon der Informatik, Heidelberg, Luzern, 2010[15], https://epdf.pub/lexikon-der-informatik-15-auflage.html, Stand: 31.07.2019, S. 626.

3 Konstantin Baierer, Philipp Zumstein,Verbesserung der OCR in digitalen Sammlungen von Bibliotheken, in: Zeitschrift für Bibliothekskultur, Mannheim, Nr. 2, 2016, S. 72–83, Netz: https://madoc.bib.uni-mannheim.de/41442/1/Baierer-Zumstein-2016.pdf, Stand: 10.08.2019, S. 80.

Schaffung einer digitalen Bibliothek hat bisher mehr als 25 Millionen Bücher[4] bei einem weltweit geschätzten Bestand von ca. 130 Millionen Titeln[5] abgespeichert. Das Münchner Digitalisierungszentrum stellt derzeit circa 2,5 Millionen Werke für die Nutzer bereit.[6]

Eine Digitalisierung von Archivgut steht im deutschsprachigen Raum unterdessen erst am Anfang.[7] Lediglich Einzelprojekte im Bereich des Editionswesens wagen sich an diese Herausforderung bereits seit längerer Zeit heran. So existieren Nachlasseditionen herausragender Persönlichkeiten in digitaler Form wie im Falle des Philosophen Ludwig Wittgenstein. Die Wittgenstein Archives (WAB, künftig Wittgensteinarchiv) in Bergen haben es sich zum Ziel gemacht, die über mehrere Institutionen und Länder verstreuten Dokumente unter einem Dach digital wiederzuvereinigen und einem weltweiten Publikum zur Verfügung zu stellen.[8]

Die Archivalien wurden in Ermangelung zuverlässiger Texterkennungssysteme seit 1990 manuell transkribiert. Ergebnis der Arbeiten war im Jahr 2000 zunächst die Bergen Electronic Edition (BEE).[9] Sie konnte auf mehreren CD-Roms erworben werden und war ausschließlich unter Microsoft Windows lauffähig. Erst deutlich später wurde mit der Übernahme und Erweiterung der BEE-Transkriptionen in den Netzauftritt „Wittgenstein-Source“[10] ein plattformübergreifender, öffentlicher Zugang geschaffen.

Das Projekt fand über die Philosophie hinaus Beachtung. So auch am Centrum für Informations- und Sprachverarbeitung (CIS) der LMU in München. Im Rahmen einer Gemeinschaftsarbeit unter der Leitung von Dr. Maximilian Hadersbeck entstand die Suchmaschine „WiTTFind“,[11] die neue

---

4 Stephen Heyman, Google Books: A Complex and Controversial Experiment, in: The New York Times, New York, 28.10.2015, Netzausgabe: https://www.nytimes.com/2015/10/29/arts/international/google-books-a-complex-and-controversial-experiment.html, Stand: 31.07.2019.

5 Leonid Taycher, Books of the world, Stand: up and be counted! All 129,864,880 for you, in: Google Books Search, [o. O.], 05.08.2010, Netz: https://booksearch.blogspot.com/2010/08/books-of-world-stand-up-and-be-counted.html, Stand: 10.08.2019, Joab Jackson, Google: 129 Million Different Books Have Been Published, in: PCWorld, [o.O], 06.08.2010, Netzausgabe: https://www.pcworld.com/article/202803/google_129_million_different_books_have_been_published.html, Stand: 10.08.2019.

6 Münchner Digitalisierungszentrum, Netzauftritt: https://www.digitale-sammlungen.de/, Stand: 14.08.2019.

7 Günter Mühlberger, Die automatisierte Volltexterkennung historischer Handschriften als gemeinsame Aufgabe von Archiven, Geistes- und Computerwissenschaftlern. Das Modell einer zentralen Transkriptionsplattform als virtuelle Forschungsumgebung, in: Irmgard Christa Becker (Hg.), Stephanie Oertel (Hg.), Digitalisierung im Archiv. Neue Wege der Bereitstellung des Archivguts. Beiträge zum 18. Archivwissenschaftlichen Kolloquium der Archivschule Marburg, Marburg an der Lahn, 2015, S. 87-115, Netz: https://www.academia.edu/7451967/Die_automatisierte_Volltexterkennung_historischer_Handschriften_als_gemeinsame_Aufgabe_von_Archiven_Geistes-_und_Computerwissenschaftlern._Das_Modell_einer_zentralen_Transkriptionsplattform_als_virtuelle_Forschungsumgebung, Stand: 10.08.2019.

8 [O. A.], The Wittgenstein Archives at the University of Bergen (WAB), Bergen in Norwegen, Netz: http://wab.uib.no/, Stand: 10.08.2019.

9 [O. A.], Wittgenstein's Nachlass. The Bergen Electronic Edition (BEE), Bergen in Norwegen, Netz: http://wab.uib.no/wab_BEE.page, Stand: 10.08.2019.

10 Wittgensteinsource, Netzauftritt: http://www.wittgensteinsource.org/, Stand: 10.08.2019.

11 WiTTFind, Netzauftritt: http://wittfind.cis.uni-muenchen.de/, Stand: 10.08.2019.

Möglichkeiten in Sachen Durchsuchbarkeit, Lesbarkeit und Vergleichbarkeit eröffnet und dem Nutzer darüber hinaus eine tiefgehende linguistische Analyse der Dokumente bietet.

Letztendlich ist eine Edition, so hochwertig sie auch sein mag, immer nur eine mehr oder minder subjektive Reproduktion der originalen Texte und kann diese nicht ersetzen. Sie ist nie gänzlich frei von Fehlern, die „Aura des Originals“[12] wird trotz der besten Transkription nicht überflüssig. Es erschien folglich besonders reizvoll auch fotografische Aufnahmen in Form eines Faksimilereaders[13] zu integrieren und Suchergebnisse von WiTTFind mit zugehörigen Bildausschnitten zu verknüpfen, um dem Nutzer einen noch direkteren Zugang zu ermöglichen.[14]

Zur Extraktion der erforderlichen Bildkoordinaten bot sich der Einsatz von OCR-Technologie an. Zwar besteht der Nachlass überwiegend aus Manuskripten. Das Vorhandensein einer akkuraten Transkription wie auch hochwertiger Faksimile ließen Texterkennungssysteme jedoch als mögliches Werkzeug erscheinen, um beide Datenbestände miteinander in Verbindung zu bringen.

Anliegen der vorliegenden Arbeit ist es, die dabei aufgetretenen Probleme vorzustellen und erarbeitete Lösungen aufzuzeigen. Folgende Fragen sollen im Mittelpunkt der Betrachtung stehen: Ist OCR-Technologie ein geeignetes Werkzeug zur Verknüpfung von digitaler Edition und Faksimile? Wie sind die vorhanden Daten zur Verarbeitung aufzubereiten? Welche OCR-Anwendung eignet sich warum für die Bearbeitung des Datenbestandes am besten? Ist es möglich, mit vorhandenen Systemen Vorschläge zu erzeugen, die eine menschliche Nachkorrektur entbehrlich machen? Können die Daten aus der Edition zum Training eines Handschriftmodells herangezogen werden?

Zunächst soll ein einführender Überblick über Ludwig Wittgenstein und seinen Nachlass gegeben werden. Daran anschließend umreißt der folgende Abschnitt die verschiedenen OCR-Arbeitsschritte und stellt vorhandene Anwendungen und deren Möglichkeiten in groben Zügen vor.

Auf dieser Basis möchte die Arbeit in einem dritten Teil die für WiTTFind entwickelten Lösungen präsentieren und darlegen, welche Anwendungen warum ausgewählt wurden. Hierbei kommen die einzelnen Schritte von der Vorbearbeitung der digitalen Faksimile bis hin zur Nachkorrektur der errechneten Bildkoordinaten zur Sprache. Abschließend soll evaluiert werden, inwieweit die erarbeiteten Werkzeuge ihre Aufgabe erfüllen, wo im momentanen Zustand Schwächen liegen, wie die vorhandene Software weiter verbessert werden kann und den Daten ein größerer Mehrwert zu

12 Vgl. hierzu: Klaus Ceynowa, Von der Aura des Originals zur Immersivität des Digitalen Experimente der Bayerischen Staatsbibliothek im virtuellen Kulturraum, in: Bibliotheken. Innovation aus Tradition, Klaus Ceynowa (Hg.), Martin Hermann (Hg.), Berlin, München, 2014, S. 249-257, Netz: https://www.degruyter.com/view/books/9783110310511/9783110310511.249/9783110310511.249.xml, Stand: 31.07.2019.

13 Faksimile Reader, Netzauftritt: http://reader.wittfind.cis.lmu.de/, Stand: 10.08.2019.

14 Günter Mühlberger, Die automatisierte Volltexterkennung, Marburg, 2015, S. 100f.

eigen ist. Dabei wird erörtert, ob die entwickelten Werkzeuge sich auch auf andere Datenbestände übertragen lassen. Dieses Vorgehen erlaubt eine schrittweise Annäherung an die eigentliche Kernfrage der Arbeit zur Verknüpfung von digitaler Edition und Faksimile. Zugleich ist eine Heranführung an die tangierten Themenbereiche gegeben.

Zur Optischen Zeichenerkennung steht im Netz überwiegend englischsprachige Fachliteratur bereit. Ausgewertet wurden Workshopmaterialien,[15] Netzauftritte,[16] Benutzerhandbücher,[17] Bedienungsanleitungen[18] und Präsentationsfolien.[19] Eine zusammenfassende Überblicksdarstellung zur Thematik Texterkennung existiert bisher nicht.

Die quelloffene OCR-Welt lässt sich in die Bereiche Ocropus zuzüglich Ableger, Tesseract und übrige OCR-Lösungen einteilen. Um den Umfang einer Bachelorarbeit nicht zu sprengen beschränken sich die Ausführungen auf Ocropus und Tesseract.

Den aktuellen technischen Stand zu Ocropus repräsentiert ein Aufsatz von Christoph Wick, Christian Reul und Frank Puppe zum Ablegersystem Calamari aus dem Jahr 2018.[20]

Lediglich relativ alte Publikationen gibt es über Tesseract. Darunter ein Beitrag von Ray Smith, der die technischen Grundlagen der Zeichenerkennung gut zusammenfasst.[21] Zu weiteren Merkmalen ist ein vom gleichen Urheber veröffentlichter umfangreicher Foliensatz im Netz abrufbar.[22] Die neueste Version vier befindet sich noch im Beta-Stadium und ist abgesehen von der üblichen Dokumentation noch nicht in eine Darstellung eingeflossen.

Die Ausführungen zum Thema Handschrifterkennung, das ebenso zur Vorstellung kommt, beziehen sich im Wesentlichen auf das „READ“-Projekt der Europäischen Union und seine Plattform Transkribus. Die wichtigsten Informationen hierzu stammen aus einem Aufsatz von Günter

15 Florian Fink, Uwe Springmann, CIS OCR Workshop v1.0. OCR and postcorrection of early printings for digital humanities, München, 2016, Netz: https://github.com/cisocrgroup/OCR-Workshop, https://zenodo.org/record/46571, Stand: 10.08.2019.

16 OCR-D, Koordinierte Förderinitiative zur Weiterentwicklung von Verfahren der Optical Character Recognition (OCR), Netzauftritt: http://ocr-d.de/, Stand: 10.08.2019.

17 ScanTailor, GitHub-Repositorium, Netz: https://github.com/scantailor/scantailor/wiki/User-Guide, Stand: 10.08.2019.

18 [O. A.], How to train a HTR model in Transkribus, Netz: https://transkribus.eu/wiki/images/3/34/HowToTranscribe_Train_A_Model.pdf, Stand: 10.08.2019.

19 Stefan Weil, Philipp Zumstein, Mit freier Software Text in Digitalisaten erkennen. OCR-Praxis an der UB Mannheim, Mannheim, 2016, Netz: https://www.kitodo.org/fileadmin/groups/kitodo/Dokumente/UBMannheim_OCR-Freie-Software.pdf, Stand: 10.08.2019.

20 Frank Puppe, Christian Reul, Christoph Wick, Calamari. A High-Performance Tensorflow-based Deep Learning Package for Optical Character Recognition, Würzburg, 2018, Netz: www.arxiv.org/pdf/1807.02004, Stand: 10.08.2019.

21 Ray Smith, An overview of the Tesseract OCR Engine in: Proc. 9th IEEE Intl. Conf. on Document Analysis and Recognition, [o. O.], 2007, S. 629–633, Netz: https://static.googleusercontent.com/media/research.google.com/de//pubs/archive/33418.pdf, Stand: 10.08.2019.

22 Ray Smith, What You Always Wanted To Know About Tesseract, [o. O.], 2016, Netz: https://github.com/tesseract-ocr/docs/tree/master/das_tutorial2016, Stand: 31.07.2019.

Mühlberger aus dem Jahr 2015[23] sowie einer umfassenden Gemeinschaftsveröffentlichung von 2019.[24] Wertvolle Anregungen haben die Teilnahme an mehreren Workshops der sechsten DHD-Tagung,[25] Fernkonferenzen sowie Arbeitstreffen in Bergen, Halle und Innsbruck ergeben.

Die Informationen zu WiTTFind wurden im Austausch mit den verantwortlichen Entwicklern, dem Quelltext verschiedener Git-Repositorien,[26] Plakaten und Postern[27] sowie dem Netzauftritt[28] entnommen. Die Daten der Edition wie auch die Faksimilebilder hat das Wittgensteinarchiv freundlicherweise bereitgestellt. Zur Person Wittgensteins lieferte der entsprechende Artikel im Wiener Geschichtslexikon[29] die wichtigsten Informationen.

23 Günter Mühlberger, Die automatisierte Volltexterkennung, Marburg, 2015.

24 Günter Mühlberger, Innsbruck, 2019, H2020 Project READ. Recognition and Enrichment of Archival Documents. 2016-2019, Netz: https://www.academia.edu/22653102/H2020_Project_READ_Recognition_and_Enrichment_of_Archival_Documents_-_2016-2019, Stand: 10.08.2019.

25 Digital Humanities im deutschsprachigen Raum e.V. (Hg.), DHd 2019 Digital Humanities: multimedial & multimodal. Konferenzabstracts, Frankfurt am Main 2019, Netz: https://zenodo.org/record/2596095, Stand: 10.08.2019.

26 GIT-Lab CIS: https://gitlab.cis.uni-muenchen.de, Stand: 10.08.2019.

27 Ines Röhrer, Sabine Ullrich, Weltkulturerbe international digital. Erweiterung der Wittgenstein Advanced Search Tools durch Semantisierung und neuronale maschinelle Übersetzung, München, 2019, Netz: https://www.cis.uni-muenchen.de/kurse/max/homepage/poster/posterdhd_2019.pdf, Stand: 10.08.2019.

28 WiTTFind, Netzauftritt: http://wittfind.cis.uni-muenchen.de/, Stand: 10.08.2019.

29 [O. A.], Ludwig Wittgenstein, in: Wien Geschichte Wiki, Netz: https://www.geschichtewiki.wien.gv.at/Wien_Geschichte_Wiki?curid=11436, Stand: 31.7.2019.

2. Ludwig Wittgenstein und sein Nachlass

Ludwig Wittgenstein wurde am 26. April 1889 im Wiener Bezirk Neuwaldegg als jüngstes von acht Kindern geboren. Die Familie gehörte zur Wirtschaftselite der k. und k. Doppelmonarchie Österreich-Ungarn. Der junge Ludwig studierte seit 1908 in Berlin Physik und wandte sich nach erfolgreichem Hochschulabschluss in London dem Thema Flugmechanik zu. 1911 wechselte Wittgenstein nach Cambridge und entdeckte dort unter seinem Förderer Bertrand Russell die Philosophie als seine eigentliche Berufung. 1913 zog sich der angehende Philosoph in die Abgeschiedenheit der norwegischen Fjorde zurück, um an seiner Dissertation zu arbeiten.

Während seinem Einsatz an der Front im damaligen Kronland Galizien vollendete Wittgenstein seine logisch-philosophische Abhandlung, die als Schlüsselwerk der Philosophie gilt. Nach Kriegsende verzichtete er auf seinen Anteil am väterlichen Erbe und war einige Jahre als Volksschullehrer tätig. 1929 verließ Wittgenstein seine Heimat Österreich, um sich in Cambridge vollends der Philosophie zuzuwenden. Von 1939 bis 1947 hatte er den dortigen Lehrstuhl für Philosophie inne. Bis zu seinem Tod im Jahre 1951 arbeitete der Wissenschaftler weiter intensiv an seinen Theorien und Überlegungen.[30]

Den Nachlass vermachten testamentarisch eingesetzte Nachlassverwalter zu großen Teilen den Universitäten Cambridge und Oxford sowie der Wiener Nationalbibliothek. Bedeutende Dokumente aus dem Bestand wurden seither in verschiedenen Editionsprojekten der Öffentlichkeit zugänglich gemacht. Die Schriftstücke sind in Typo- und Manuskripte eingeteilt und entsprechend mit den Kürzeln Ms bzw. Ts sowie laufenden Nummern versehen. Zur weiteren Strukturierung des Materials entstanden Siglen, die sich aus dem Dokumentnamen, der Seite und einer Bemerkungsnummer zusammensetzen. Sie referenzieren eindeutig verschieden lange, inhaltlich zusammengehörige Textabschnitte und bilden die zentralen Strukturelemente der digitalen Edition. Die Siglen können dabei eine ganze Seite umfassen. Mehrfachsiglen können mehrere Seiten miteinander verknüpfen.

Eine manuelle Transkription und elektronische Speicherung erlaubte es, die verschiedenen Teile der Hinterlassenschaft wiederzuvereinigen und weltweit zugänglich zu machen. Die Faszination für die Philosophie Wittgensteins ist ungebrochen. 2017 wurde der Nachlass von der UNESCO zum Weltdokumentenerbe erhoben.[31] Der Bestand umfasst etwa 20.000 überwiegend handschriftlich verfasste Seiten, zu denen jeweils eine digitale Fotografie existiert. Circa 5.000 der Aufnahmen sind gemeinfrei verwendbar. Die restlichen 15.000 Digitalisate unterliegen dem Urheberrecht der Dokumenteigner, dürfen jedoch in WiTTFind dargestellt werden.

30 Ebd.

31 [O. A.], The Wittgenstein Archives, Bergen, Netz: http://wab.uib.no/, Stand: 10.08.2019.

3. OCR-Anwendungen

3.1. Einführung: Die OCR-Verarbeitungskette

Die digitale Fotografie hat auch die OCR-Technologie auf eine neue Grundlage gestellt. In ihrer Entwicklung wurde eine unüberschaubar große Zahl an Geräten, Anwendungen und Programmen geschaffen, angefangen bei elektrooptischen Apparaten.[32] Mit dem breiten Aufkommen von digitalen, freiprogrammierbaren Rechnern hat sich die optische Erkennung ganz auf diesen Bereich verlagert. Heute existieren eine Vielzahl an Texterkennungssystemen für verschiedene Plattformen. Sie decken höchst unterschiedliche Teilbereiche des Verarbeitungsprozess von der Retrodigitalisierung über die Vorverarbeitung, die Binarisierung, die eigentliche Texterkennung bis hin zur Nachbearbeitung ab. In der vorliegenden Arbeit können wie eingangs angesprochen nur die leistungsfähigsten und verbreitetsten Vertreter aus dem quelloffenen Bereich der Texterkennungssysteme Tesseract, Ocropus und seine Ableger sowie das Vorverarbeitungswerkzeug ScanTailor[33] umrissen werden.

3.2. Vorbearbeitung am Beispiel ScanTailor

ScanTailor ermöglicht mit Hilfe einer grafischen Benutzeroberfläche einen komplett geführten Arbeitsdurchlauf zur Stapelvorverarbeitung ausgehend von Ausgabebildern des Einlesegerätes bzw. der Kamera bis hin zum ausgerichteten, zugeschnittenen und bereinigten Binärbild.

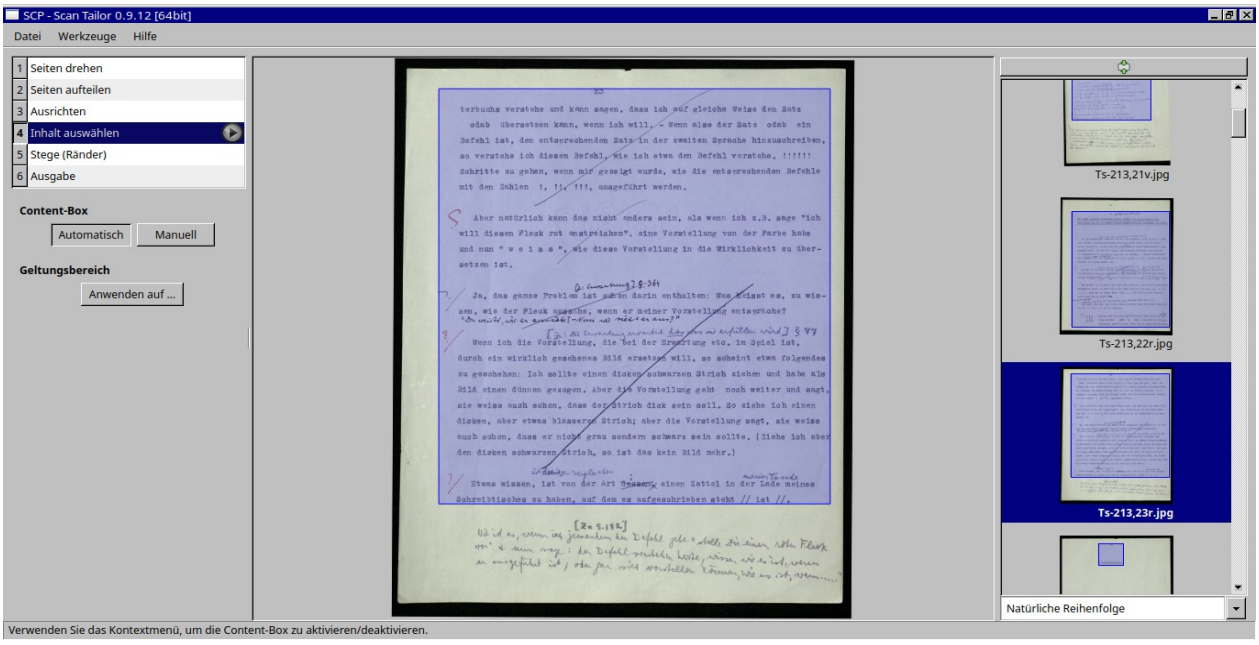


Abb. 1: Grafische Benutzeroberfläche von ScanTailor, Beispiel mit Fehlsegmentierungen (Ts-213)

32 Michael Day, IMPACT best practice guide: Metadata for text digitisation and OCR, [o. O.], 2010, Netz: https://www.digitisation.eu/download/website-files/BPG/OpticalCharacterRecognition-IBPG_01.pdf, Stand: 31.07.2019, S. 1-4.

33 ScanTailor, Netzauftritt, https://scantailor.org/, Stand: 10.08.2019.

Letzteres kann unmittelbar an die OCR-Anwendung übergeben werden. Als Eingabe akzeptiert ScanTailor ein ganzes Verzeichnis mit Bilddateien. In sechs aufeinanderfolgenden Arbeitsschritten werden die Bilder so modifiziert, dass als Ergebnis für die OCR-Anwendung gleich große und fertig aufbereitete Seiten zur Verfügung stehen. Durch die Ermittlung eines Textbereiches auf der Seite werden störende Ränder entfernt und die Seiten auf einen einheitlichen Rahmen gespannt. Gerade im Fall von Verzerrungen im Bild, bei Verschmutzungen auf der Seite oder bei Wölbungen des Papiers kann ScanTailor ein wichtiges Hilfsmittel darstellen, um die Ausgangsdateien und damit auch das Endergebnis des gesamten Erkennungsvorgangs entscheidend zu verbessern. Über die grafische Oberfläche wird der Anwender in die Lage versetzt, jeden Schritt zu prüfen, die automatisch generierten Vorschläge zu korrigieren und zahlreiche Parameter wie den Schwellenwert bei der Binarisierung anzupassen. Dabei ist für jede einzelne Eingabedatei eine besondere Feineinstellung möglich.

Weiterhin existiert ein eigenes Kommandozeilenwerkzeug.[34] Es bietet allerdings weitaus weniger Feinregulierungsoptionen. ScanTailor findet breite Anwendung bei der Retrodigitalisierung. Bedauerlicherweise ist die Fortentwicklung ins Stocken geraten. Die letzte Aktualisierung wurde 2016 eingespielt, die letzte stabile Version stammt aus dem Jahr 2012.[35] ScanTailor Advanced[36] und ScanTailor Enhanced[37] bieten als Ablegerprojekte einen erweiterten Funktionsumfang, müssen allerdings aus dem Quelltext gebaut werden. Die Neuimplementierung in Python wie die Erweiterung der Kommandozeilenfunktionalität wären für künftige Vorhaben von großem Nutzen.[38]

3.3. Quelloffene OCR-Anwendungen

3.3.1. Tesseract

Eine längere Entwicklungspause gibt es auch in der Historie von Tesseract. Die OCR-Anwendung wurde 1984 im Rahmen einer Doktorarbeit bei „Hewlett-Packard“ (HP) in der Programmiersprache C++ entwickelt. Das Programm kam bei zahlreichen Einlesegeräten zum Einsatz und wurde bis 1994 stetig weitergepflegt.[39] In der Folgezeit verfolgte HP das Projekt nicht länger und der Quelltext blieb unverändert. 2005 erwarb Google die Rechte und legte den „Code“ offen. In der Folge wurde das System grundlegend überarbeitet und unter anderem mit einem Modul zur vollautomatischen

34 ScanTailor, GitHub-Repositorium, Netz: https://github.com/scantailor/scantailor/blob/master/CommandLine.cpp, Stand: 10.08.2019.
35 Ebd.
36 ScanTailor Advanced, GitHub-Repositorium, Netz: https://github.com/4lex4/scantailor-advanced, Stand: 10.08.2019.
37 scantailor_enhanced, GitHub-Repositorium, Netz: https://gist.github.com/tristan-k/2700275, Stand: 10.08.2019.
38 Diskussion in der AGOCR am 14. Juni 2019.
39 Ray Smith, An overview, 2007, S. 1, Victor Ohlsen, Optical Character and Symbol Recognition using Tesseract, Luleå, 2016, S. 8, Netz: http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1019846&dswid=2838, Stand: 10.08.2019.

Seitensegmentierung ausgestattet, welches auf der Bildverarbeitungsbibliothek Leptonica aufsetzt.[40] Im gleichen Zug konnte eine Unterstützung der gängigsten Bildformate als Eingabedatei realisiert werden. Auch die direkte Verarbeitung von Farb- und Graustufenbildern kam zum standardmäßigen Funktionsumfang hinzu.

Tesseract arbeitet mit optischer Mustererkennung auf Grundlage einzelner disjunkter Zeichen. Für jede Glyphe muss eine entsprechende Schablone vorhanden sein. Das für die jeweils ähnlichste Schablone codierte Zeichen wird zurückgegeben.[41]

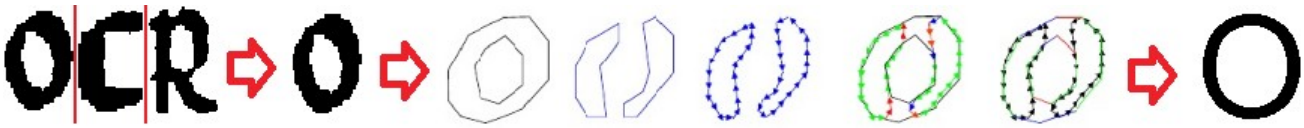


Abb. 2: Erkennungsschritte für den Buchstaben „O“ mit Tesseract

Es kommen Wörterbücher wie auch Sprachmodelle zum Einsatz, um die Ausgabe zu verbessern.[42] Für die Eingabetexte sind entsprechende Grundlagen eine Voraussetzung zur Erkennung. Ab der Version vier, die das Betastadium bisher nicht verlassen hat, wird zusätzlich ein auf neuronalen Netzen basierender Identifikator eingesetzt.[43] Er verbessert die Ergebnisse teilweise deutlich.[44]

Praktisch alle Bildformate mit Ausnahme von PDF werden als Eingabe akzeptiert. Zur Stapelverarbeitung eignen sich am besten mehrseitige TIF-Dateien, die Tesseract durchläuft und das Resultat in eine einzige Ausgabedatei schreibt. Als Zielformate werden standardmäßig PDF, TXT und HOCR angeboten, künftig wird Ausgabe von ALTO-Dateien möglich sein.[45] Das Modul TesseractOCRToPAGE[46] erlaubt eine direkte PAGE-XML-Ausgabe. Die Anwendung unterstützt damit alle gängigen Ausgabeformate und bildet die Grundlage für zahlreiche OCR-Programme mit grafischer Benutzeroberfläche wie beispielsweise den gImageReader.[47] Tesseract wird unter Unix-Systemen durch einen gleichnamigen Befehl auf der Kommandozeile aufgerufen und der Erkennungsvorgang gestartet.

tesseract <Eingabedatei> <Ausgabedatei> [-l <Sprache>] [<Ausgabeformat>]

Abb. 3: Aufruf von Tesseract

40 Victor Ohlsen, Optical Character and Symbol Recognition using Tesseract, Luleå, 2016, S. 8.
41 Ebd. 12-15.
42 Frank Puppe, Christian Reul, Uwe Springmann, Christoph Wick, State of the Art Optical Character Recognition of 19th Century Fraktur Scripts using Open Source Engines, Würzburg, 2018, Netz: https://arxiv.org/abs/1810.03436, Stand: 10.08.2019, S. 2.
43 Ebd.
44 Nach wie vor werden fehlerhafte oder leere Ausgaben für einige Eingabeseiten erzeugt.
45 Tesseract-OCR, GitHub-Repositorium, Netz: https://github.com/tesseract-ocr/tesseract/blob/master/README.md, Stand: 10.08.2019.
46 Pattern Recognition & Image Analysis Reasearch Lab, Universität Salford, Manchester, Netzauftritt: https://www.primaresearch.org/tools/TesseractOCRToPAGE, Stand: 10.08.2019.
47 GImageReader, GitHub-Repositorium, Netz: https://github.com/manisandro/gImageReader, Stand: 10.08.2019.

Für Feineinstellungen besteht die Möglichkeit, mithilfe von Skripten integrierte Teilschritte wie die Seitensegmentierung zu konfigurieren. Außerdem werden zusätzlich direkt über die Kommandozeile ansprechbare Parameter wie zur Einstellung Sprache angeboten. Tesseract unterstützt mehr als 100 Sprachen und stellt zudem Sondermodule zum Beispiel für deutsche Frakturdrucke zur Verfügung.[48] Die Modelle sind in der Lage, eine relativ große Bandbreite an Dokumenten richtig bzw. weitgehend richtig zu erkennen. Die Ausbildung eigener Modelle ist grundsätzlich möglich, jedoch mit sehr großem Aufwand verbunden und nur für professionelle Bearbeiter von Interesse.[49]

3.3.2. Von Ocropus bis Calamari

Ganz auf den Ansatz des Endnutzertrainings und der Erkennung mittels neuronaler Netze setzen unterdessen Ocropus[50] und seine zahlreichen Ableger und Versionen, darunter Ocropy,[51] Kraken[52] oder Calamari.[53] Es werden weder Wörterbücher noch Sprachmodelle verwendet.[54] Das System ist lediglich abhängig von der zugrundeliegenden Schrift. Die Urvariante Ocropus wurde von Thomas Breuel seit 2006 am Deutschen Forschungsinstitut für künstliche Intelligenz in Zusammenarbeit mit Google in C++ entwickelt und 2008 veröffentlicht.[55] Mittlerweile ist im Projekt Ocropy der Quelltext in Python implementiert.[56]

Ocropus liefert einen ganzen Werkzeugkasten an einzelnen Befehlen, die verschiedene Aufgaben der Verarbeitungskette vom Quellbild zur Endausgabe übernehmen. So sind die Binarisierung, die Segmentierung und die eigentliche Erkennung jeweils getrennt auszuführen.[57]

Revolutionär war eine Abkehr von der zeichenweisen Erkennung wie in den meisten bisherigen Anwendungen üblich. Ocropus verfolgt stattdessen einen zeilenbasierten Ansatz. Über das Zeilenbild wird von links nach rechts ein ein-Pixel breiter Rahmen geschoben. Die Ausprägung und die Abfolge der auftretenden Pixelmuster ermöglicht der Maschine die Erkennung verschiedener Buchstaben. Auf diesem Weg sind Kontexte und theoretisch auch Ligaturen unterscheidbar.

48 Tesseract-OCR, GitHub-Repositorium, Netz: https://github.com/tesseract-ocr/tesseract/blob/master/README.md, Stand: 10.08.2019.
49 Tesseract-OCR, GitHub-Repositorium, Netz: https://github.com/tesseract-ocr/tesseract/wiki/TrainingTesseract, Stand: 10.08.2019.
50 Ocropus, GittHub-Repositorium, Netz: https://github.com/jkrall/ocropus/tree/master/ocropus, Stand: 10.08.2019.
51 Ocropy, GitHub-Repositorium, Netz: https://github.com/tmbdev/ocropy, Stand: 10.08.2019.
52 Kraken, GitHub-Repositorium, Netz: https://github.com/mittagessen/kraken, Stand: 10.08.2019.
53 Calamari-OCR, GitHub-Repositorium, Netz: https://github.com/Calamari-OCR, Stand: 10.08.2019.
54 Frank Puppe, State of the Art Optical Character Recognition, Würzburg, 2018, S. 2.
55 Ebd.
56 Ocropy, GitHub-Repositorium, Netz: https://github.com/tmbdev/ocropy, Stand: 10.08.2019.
57 Ebd., vgl. z.B. die Programme ocropus-nlbin, ocoropus-gpageseg oder ocropus-rpred.

Abb. 4: Erkennung einer Zeile durch Ocropus

Ocropus ist für kleinere Datenbestände und Spezialfälle wie historische Drucke ausgelegt, für die die gelieferten Komponenten mit relativ geringem Aufwand an die jeweiligen Bedürfnisse angepasst werden können. Dafür wird ein Trainingsskript zur Erzeugung eigener Modelle mitgeliefert, welches ein automatisiertes Training ermöglicht.

Ausgegeben werden aufgrund der zahlreichen Ableger, Erweiterungen und Versionen praktisch alle Ausgabeformate. Auf der Basis von Ocropus sowie den Nachfolgern Ocropy, Ocropus zwei[58] und drei[59] aufbauend wurde das System Kraken-OCR[60] entworfen. Es setzt mehrschichtige rekurrente neuronale Netze ein und unterstützt Mehrprozessorbetrieb. Bemerkenswert ist die vereinfachte Parametrisierung und das bereitgestellte Segmentierungsmodul, welches eine Ergebnisspeicherung im JSON-Format erlaubt. Kraken hält sich weitgehend an die Befehlsstruktur von Ocropus.

Durch die Trennung der verschiedenen Arbeitsschritte ist es relativ einfach möglich, einzelne Komponenten untereinander auszutauschen oder neu zu implementieren. So gibt es für Calamari bisher nur eigene Befehle für die Erkennung und das Modelltraining. Die Binarisierung und Segmentierung wurde von Ocropus übernommen.[61] Calamari nutzt wie Kraken mehrschichtige neuronale Netze und ermöglicht darüber hinaus mit verschiedenen Voting-Algorithmen die Erkennung über mehrere Modelle. Abweichende Zwischenergebnisse sind automatisiert über Konfidenzwerte selektierbar. Zudem wird mit der Einbindung von Tensor Flow ein beschleunigtes Training wie auch die Verwendung von Grafikprozessoren für den OCR-Vorgang erreicht.[62]

Gerade die grundsätzlich über die Kommandozeile zu startenden Einzelbefehle sind eine oftmals zu hohe Hürde für viele Geisteswissenschaftler oder gewöhnliche Endanwender. Um einem größeren Nutzerkreis die Verwendung von Ocropus-basierten Werkzeugen zu erschließen wurde an der

58 Ocropus2, GitHub-Repositorium, Netz: https://github.com/tmbdev/ocropy2, Stand: 10.08.2019.
59 Ocropus3, GitHub-Repositorium, Netz: https://github.com/NVlabs/ocropus3, Stand: 10.08.2019.
60 Kraken, GitHub-Repositorium, Netz: https://github.com/mittagessen/kraken, Stand: 10.08.2019.
61 Calamari-OCR, GitHub-Repositorium, Netz: https://github.com/Calamari-OCR, Stand: 10.08.2019.
62 Ebd., Frank Puppe, State of the Art Optical Character Recognition, Würzburg, 2018, S. 1f.

Universität Würzburg ein Docker-Container entworfen, der den gesamten Arbeitsablauf auf Grundlage von Calamari in einer netzbasierten grafischen Benutzeroberfläche kapselt.[63]

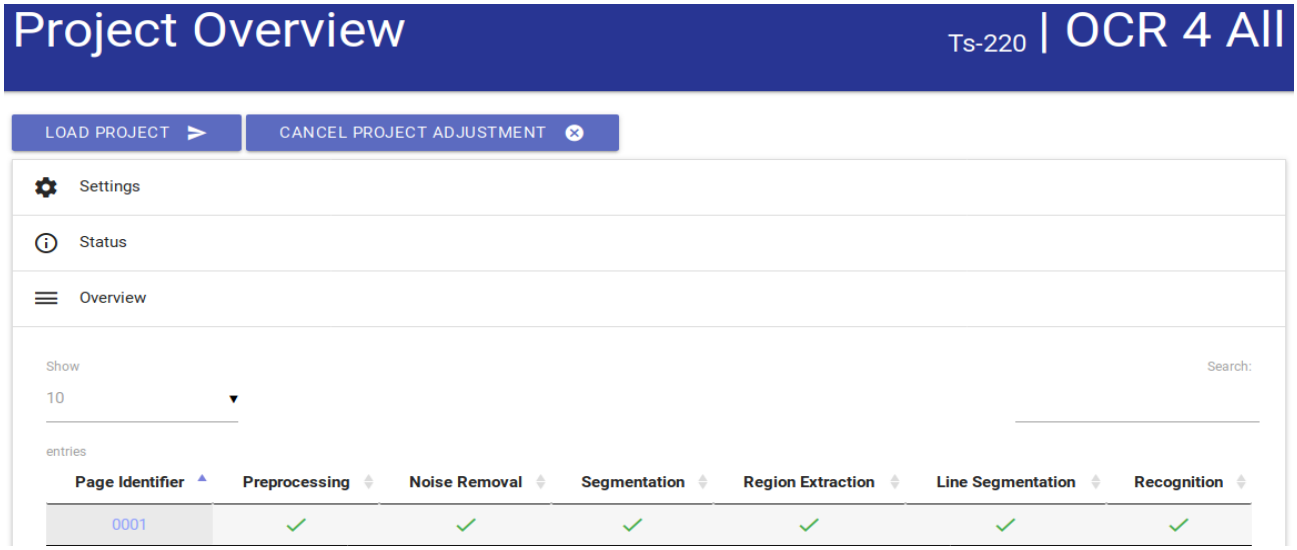


Abb. 5: OCR4all-Oberfläche

Die Ausbildung von eigenen Modellen ist dabei eingeschlossen. Mit OCR4all wird der gewöhnliche Benutzer in die Lage versetzt, mit geringem Vorwissen hochwertige OCR-Resultate zu generieren. Bei geeignetem Ausgangsmaterial können mit relativ geringem Aufwand neue Drucktypen erlernt werden. Weil kein Sprachmodell erforderlich ist, genügt ein ausreichender[64] Datenbestand an Zeilen mit korrekter Transkription um das zur Erkennung benutzte neuronale Netz zu trainieren.

3.4. GroundTruth und Training

Alle trainierbaren OCR-Anwendungen benötigen zur Ausbildung auf Eingabedokumenten mit unbekannten Schriftarten oder Sprachen entsprechendes „Lehrmaterial“, welches als „Grundwahrheit“ oder „GroundTruth“ bezeichnet wird. Dabei handelt es sich um manuell transkribierte Seiten, Zeilen oder Zeichen, die von der Anwendung als Muster behandelt werden. Je nach OCR-Lösung und Varianz des Bestandes ist mehr oder weniger Lehrmaterial erforderlich.

Ein grundsätzliches Problem ist die in Abhängigkeit vom Format und Erkennungsart beschränkte Austauschbarkeit von GroundTruth-Daten und Modellen. Für Tesseract ist abgesehen von einem Sprachmodell und Wörterbuch eine zeichenweise Segmentierung nötig, Ocropus verfolgt bekanntlich eine zeilenweise Segmentierung.

Für das Training verschiedener Druckerschriften stehen frei zugängliche Datenrepositorien im Netz bereit, aus denen Modelle bzw. Trainingsdaten erzeugt werden können. Einer der größten frei

63 OCR4all, GitHub-Repositorium, Netz: https://github.com/OCR4all, Stand: 10.08.2019.
64 Je nach Beschaffenheit des Datenbestandes ca. 10 bis 50 Seiten.

zugänglichen Bestände für historische Frakturdrucke ist beispielsweise das Repositorium GT4HistOCR, das eine Vielzahl verschiedener Frakturdrucke als Trainingsmaterial anbietet.[65]
Je nach Ähnlichkeit zur eigenen Drucktype können bessere oder schlechtere Ergebnisse mit aus einem Fremdbestand trainierten Modell erreicht werden. Schreibmaschinenseiten, Durchschläge und ältere Lithographien weisen beispielsweise eine höhere Varianz auf als neuere Drucke.

3.5. Handschrifterkennung am Beispiel Transkribus

Im Handschriftbereich kommt eine Abhängigkeit vom Schreiber hinzu. Praktisch für jede Schrift ist die Ausbildung eines eigenen Modells unerlässlich zur Generierung brauchbarer Ergebnisse. Durch die hohe Varianz handgeschriebener Texte ist ein weitaus größerer Trainingssatz erforderlich.
Bei Handschrift muss grundsätzlich zwischen Direkterkennung beim Schreibvorgang und einer Betrachtung aus der Retrospektive heraus unterschieden werden.[66] Während die Direktverarbeitung mittlerweile weit gediehen ist, bereitet die Erkennung geschriebenen Materials noch erhebliche Schwierigkeiten. Das Fehlen der Information, in welchem Ablauf der Text entstanden ist, zwingt zur Schaffung eigener Herangehensweisen.[67]
Schon aufgrund der zeichenbasierten Erkennung kommt Tesseract für eine solche Aufgabenstellung nicht in Frage. Bereits an der Seitensegmentierung scheitert auch eine Handschrifterkennung mithilfe von Ocropus. Zwar wäre mit korrekter Segmentierung theoretisch die Ausbildung eines Modells möglich. Dafür ist die Anwendung allerdings schon dem „FAQ“ zu Folge nicht gedacht. Stattdessen wird auf Transkribus verwiesen.[68]
Dabei handelt es sich um die zentrale Plattform zur Handschriftbearbeitung der europäischen Gemeinschaftsarbeit „READ“.[69] Beteiligt an diesem wohl fundiertesten Ansatz zur Schaffung einer zuverlässigen Handschrifterkennung sind unter anderem die Universitäten Athen, Innsbruck, Rostock und Valencia. Übergeordnetes Ziel ist letztendlich die Berechnung eines universal verwendbaren Handschriftmodells.[70]
Dafür werden Datenbestände weltweit gesammelt und auf projekteigenen Rechnern abgelegt. Der Nutzer kann nach der Erstellung eines eigenen Kontos in einer Klientanwendung manuell einen Teil seines Datenbestands transkribieren.

---

65 GT4HistOCR, Zenodo-Record, Netz: https://zenodo.org/record/1344132, Stand: 10.08.2019, weitere Bestände: ocr-gt, GitHub-Repositorium, Netz: https://github.com/cneud/ocr-gt, Stand: 10.08.2019.
66 Günter Mühlberger, Die automatisierte Volltexterkennung, Marburg, 2015. S. 87-93.
67 Günter Mühlberger, Die automatisierte Volltexterkennung, Marburg, 2015, S. 89.
68 Ocropy, GitHub-Repositorium, Netz:https://github.com/tmbdev/ocropy/wiki/FAQ, Stand: 10.08.2019.
69 READ. Recognition and Enrichment of Archival Documents, Netzauftritt: https://read.transkribus.eu/, Stand: 10.08.2019, vgl. hierzu: Günter Mühlberger, Innsbruck, 2019, H2020 Project READ, Innsbruck, 2019.
70 Günter Mühlberger, Die automatisierte Volltexterkennung, Marburg, 2015, S. 107f.

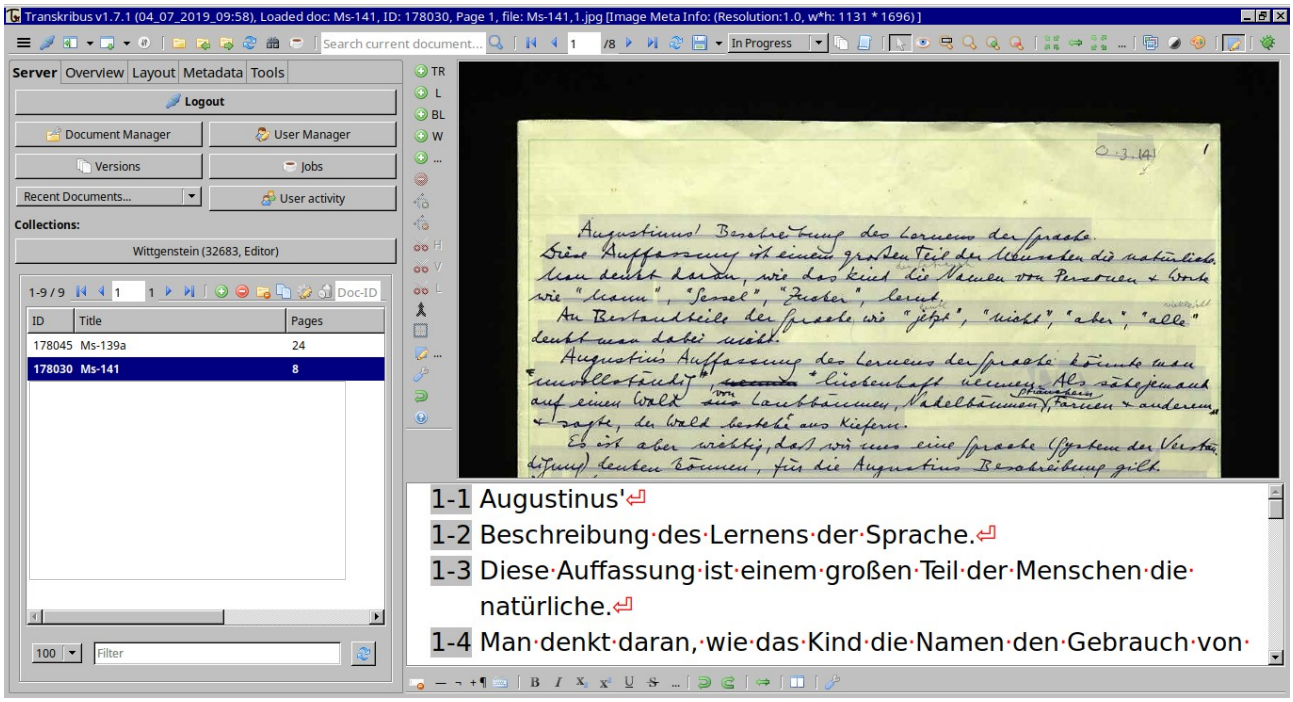


Abb. 6: Transkribus-Klient, Beispiel mit Ms-141, Fehlsegmentierung in der ersten Zeile

Nachdem eine genügend große Menge an Text in digitaler Form vorliegt, ist es möglich, ein Modell für die zugrundeliegende Handschrift auszubilden. Schon die Vorbearbeitung läuft vollständig auf dem Server und ist für den Anwender nicht einsehbar.[71]

Zwar wird Transkribus bis Jahresende von der Europäischen Union finanziert und die Nutzung ist derzeit noch kostenfrei. Der Quelltext wird jedoch unter Verschluss gehalten. Das zugrundeliegende System zur „Handwritten Text Recognition“ (HTR) benutzt ebenso eine Erkennung auf Grundlage neuronaler Netze. Mit Beginn des Jahres 2020 soll sich das Vorhaben als genossenschaftliche Unternehmung selbst tragen. Dafür ist die Einführung eines Bezahlmodells geplant, das je nach Nutzergruppe unterschiedlich ausgestaltet ist.[72]

Für die Erkennung gedruckter Texte wurde die kommerzielle Anwendung Abbyy FineReader[73] eingebunden. Der FineReader ist im Vorgängerprojekt „IMPACT“ entstanden. Seit Auslaufen der öffentlichen Finanzierung operieren mehrere Nachfolgeunternehmen gewinnorientiert mit teils erheblichen Preisen für die Endnutzer.[74]

Die Klientanwendung soll mit der Einbindung von FineReader und weiteren Anwendungen ein allumfassendes Werkzeug zur Digitalisierung bilden.[75] Die einzelnen Komponenten sind für

71 [O. A.], How to train a HTR model in Transkribus, Innsbruck, 2018, Netz: https://transkribus.eu/wiki/images/3/34/HowToTranscribe_Train_A_Model.pdf, Stand: 10.08.2019.
72 Günter Mühlberger, Innsbruck, 2019, H2020 Project READ, Innsbruck, 2019, S. 39-43.
73 Abbyy Finereader, Netzauftritt: https://www.abbyy.com/de-de/finereader/, Stand: 10.08.2019.
74 Vgl. hierzu: [o. A.], IMPACT, [o. J.], Netz: https://www.frakturschrift.com/en:projects:impact, Stand: 10.08.2019.
75 Günter Mühlberger, Die automatisierte Volltexterkennung, Marburg, 2015, S.101-108, Transkribus-Wiki, Netzauftritt: https://transkribus.eu/wikiDe/index.php/Benutzeranleitung, Stand: 10.08.2019.

einzelne Benutzer freischaltbar. Ebenso aufgenommen wurde zum Beispiel das Text2Image Werkzeug (T2I), welches an der Universität Rostock entwickelt wird.[76]

Vorhandene Transkriptionen und Bilddaten können mithilfe dieser Anwendung miteinander aligniert werden. Ohne größeren Aufwand erlaubt es T2I, vorhandene Transkriptionen und Faksimilebilder algorithmisch miteinander zu verknüpfen. Auf Basis dieser Alignierung sind Handschriftmodelle direkt trainierbar. Damit besteht die Möglichkeit, Datenbestände abgeschlossener Digitalisierungsprojekte zu nutzen. Als Eingabe werden lediglich Textdaten mit oder ohne Zeilenumbrüchen sowie zugehörige Bilddateien vorausgesetzt. Auf dieser Basis kann die Ausbildung eines Modells angestoßen werden.[77]

Nach Transkription bzw. Erkennung eines Dokuments bietet der Transkribus-Klient neben den Endnutzformaten DOCX, TXT und PDF die Generierung von ALTO-Dateien. Eine Ausgabe von PAGE-XML ist in Vorbereitung.[78]

3.6. Ausgabeformate für OCR: PAGE-XML, ALTO-XML und HOCR

Schon mehrfach wurde in den vorhergehenden Ausführungen das Thema Ausgabeformate angeschnitten. Zum besseren Verständnis sollen diese noch einmal kurz vorgestellt werden. Für den Zweck der optischen Zeichenerkennung sind eigene Dateiformate zur Speicherung der gewonnenen Informationen wie zum Beispiel der Seitengestalt unerlässlich. Die Koordinateninformation findet sich bei allen Formaten in sogenannten „Bounding-Boxes“ wieder. Darunter werden den Text umgebende Polygone verstanden, die durch Pixelkoordinaten gespeichert sind. Meist genügen zur Speicherung eines Rechteckes zwei Koordinatenpunkte bestehend aus vier Integerzahlen.

Das gerade angesprochene PAGE-XML ist recht verbreitet und legt die beim Erkennungsvorgang gewonnenen Informationen in beliebiger Tiefe redundant ab.[79] Es ermöglicht so den Abgleich einzelner Arbeitsschritte außerhalb der OCR-Anwendung. Der Dateityp wurde von der Universität Salford im Rahmen von „IMPACT“ für die Texterkennung entwickelt und ist in der Lage, jeden einzelnen Verarbeitungsschritt einschließlich der Nachbearbeitung aufzunehmen, die Ergebnisse verschiedener OCR-Anwendungen in einer Datei zu vereinigen sowie die Ergebnisse vergleich- und

76 Tobias Hodel, Gundram Leifert, Text2Image matching. How to use existing images and transcripts to produce Automated Text Recognition, Rostock, 2017, [o. A.], How to use existing transcriptions to train a Handwritten Text Recognition (HTR) model, Innsbruck, Rostock, 2018, Netz: https://transkribus.eu/wiki/images/6/6f/HowToUseExistingTranscriptions.pdf, Stand: 10.08.2019.

77 Ebd.

78 Stand: Februar 2019, Arbeitstreffen in Innsbruck.

79 [O. A.], 2016 PAGE XML Format for Page Content, Manchester, 2016, Netz: https://www.primaresearch.org/schema/PAGE/gts/pagecontent/2016-07-15/Simple%20PAGE%20XML%20Example.pdf, Stand: 10.08.2019, Sami Nousiainen, Report on File Formats for Hand-written Text Recognition (HTR) Material, Network, Helsinki, 2016, Netz: https://documents.icar-us.eu/documents/2016/12/report-on-file-formats-for-hand-written-text-recognition-htr-material.pdf, Stand: 10.08.2019.

bewertbar zu machen. Demgegenüber stehen mitunter komplexe teilweise unübersichtliche Ausgaben.

Ein weiterhin verbreitetes OCR-Format ist ALTO-XML,[80] welches im Rahmen des EU-Projekts „METAe“ entstanden ist. Die DFG empfiehlt das Format für die Textdigitalisierung sowie eine Verknüpfung mit METS.[81]

Die Textinformation wird einfach gespeichert und die Seitengestalt durch entsprechende Tags in einem übergeordneten Layout Element repräsentiert. Das Format erlaubt, aus den Daten TEI-konforme XML-Dokumente zu erzeugen.[82]

HOCR ist mit HTML vollständig kompatibel. Der Vorteil der Anlehnung an eines der gängigsten Formate wird mit der Beschränkung auf die vorgegebene Spezifikationen erkauft. So sind sämtliche OCR-spezifische Informationen in DIV- und SPAN-Elementen als Attribute abgelegt.

Es wird zwischen verschiedenen Bereichen wie Seiten, Absätzen, Wörtern und Zeilen unterschieden,[83] wobei zum Beispiel Tesseract den erkannten Text wortweise ablegt.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
 <head>
  <title></title>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
  <meta name='ocr-system' content='tesseract 3.04.01' />
  <meta name='ocr-capabilities' content='ocr_page ocr_carea ocr_par ocr_line ocrx_word'/>
</head>
<body>
  <div class='ocr_page' id='page_1' title='image "1.tif"; bbox 0 0 414 460; ppageno 0'>
   <div class='ocr_carea' id='block_1_1' title="bbox 98 95 193 109">
    <p class='ocr_par' dir='ltr' id='par_1_1' title="bbox 98 95 193 109">
     <span class='ocr_line' id='line_1_1' title="bbox 98 95 193 109; baseline 0 0; x_size
19.238094; x_descenders 5.2380953; x_ascenders 4"><span class='ocrx_word' id='word_1_1'
title='bbox 98 95 141 109; x_wconf 84' lang='deu' dir='ltr'>Hallo</span> <span
class='ocrx_word' id='word_1_2' title='bbox 148 95 193 109; x_wconf 81' lang='deu'
dir='ltr'>Welt!</span>
     </span>
    </p>
   </div>
  </div>
 </body>
</html>
```

Abb. 7: „Hallo Welt!“ in HOCR

80 ALTO XML, GitHub-Repositorium, Netz: https://github.com/altoxml, Stand: 10.08.2019, Sami Nousiainen, Report on File Formats, Helsinki, 2016.

81 Deutsche Forschungsgemeinschaft (Hg.), DFG-Praxisregeln „Digitalisierung“, Bonn, 2016, Netz: http://www.dfg.de/formulare/12_151/12_151_de.pdf, Stand: 10.08.2019, S. 37f, Diese Empfehlung befindet sich derzeit in der Überarbeitung und soll künftig um Page-XML ergänzt werden, (Fernkonferenz 26.07.2019).

82 Deutsche Forschungsgemeinschaft (Hg.), DFG-Praxisregeln „Digitalisierung“, Bonn, 2016, S. 37f.

83 HOCR-Spezifikation, Netzauftritt: http://kba.cloud/hocr-spec/1.2/, Stand: 10.08.2019.

Die eigentliche Textinformation wird jeweils nur in der spezifischsten Ebene einfach abgespeichert. Wie in den anderen Formaten ist ein Konfidenzwert vorgesehen, der dazu dient, im Rahmen einer Nachkorrektur unsicher erkannte Wörter gesondert zu behandeln.

HOCR-Daten können mit jedem Browser geöffnet und ihr Inhalt lesbar dargestellt werden. Die OCR-spezifischen Informationen bleiben in der HTML-Darstellung verborgen. Sie sind nur im Rohtext sichtbar. HOCR unterstützt wie alle anderen Formate das Abspeichern mehrerer Seiten in einer Datei. Häufig wird pro Bilddatei eine OCR-Datei erstellt.

Das Konvertieren der Formate untereinander ist nur bedingt möglich. Während ALTO-XML und HOCR theoretisch in PAGE-XML konvertiert werden kann, ist die Umwandlung in die andere Richtung schon wegen dem häufig komplexeren Aufbau nicht ohne Beachtung der speziellen Ausprägung der Dateien realisierbar. Ein Konvertieren von PAGE-XML nach PDF ist im Moment nur mit Einbindung einer kommerziellen Java Bibliothek möglich. Auch wurde die Spezifikation von Page-XML in der Vergangenheit mehrfach angepasst.[84] Ein universell konfigurierbares Konversionswerkzeug wäre in zahlreichen Anwendungsfällen eine große Hilfe. Bisher existieren lediglich fehlerbehaftete Einzellösungen, die über die Kommandozeile zu steuern sind.[85]

Alle angesprochenen Formate können in TEI-konforme Daten als Grundstock einer digitalen Edition umgewandelt werden. Diese sind im Falle des Wittgenstein-Nachlasses bereits in fertiger Form vorhanden. Für die Bedürfnisse von WiTTFind wurden nach dem Parsen von HOCR-Ausgaben einfache Datenstrukturen auf JSON-Basis zum Informationsaustausch zwischen verschiedenen Komponenten erstellt.

84 Für die Aktualisierung in das aktuelle PAGE-XML-Format existieren eigene Konverter, vgl: Pattern Recognition & Image Analysis Reasearch Lab, Universität Salford, Manchester, Netzauftritt: https://www.primaresearch.org/tools/PAGEConverterValidator, Stand: 10.08.2019.

85 OCR-conversion-scripts, GitHub-Repositorium, Netz: https://github.com/cneud/ocr-conversion-scripts, Stand: 10.08.2019, Awesome OCR, GitHub-Repositorium, Netz: https://github.com/kba/awesome-ocr#ocr-file-formats, Stand: 10.08.2019.

4. Arbeitskette für WiTTFind

4.1. Einführung – Von der Vor- zur Nachbearbeitung

Im Folgenden soll der für WiTTFind ausgearbeitete Arbeitsablauf vorgestellt werden. Zunächst kommt der Bereich der Vorbearbeitung zur Sprache. Daran anschließend wird der Einsatz von Texterkennung betrachtet. Der dritte Abschnitt handelt von der Alignierung von Edition und Faksimile sowie der Berechnung möglichst präziser Koordinatenvorschläge für die nach Siglen strukturierten anonymen Blöcke der Edition. Vorrangiges Ziel war die Ermittlung entsprechender Bildkoordinaten für jedes Siglum.

Die Bildverkleinerung wurde als erste Teilaufgabe auf dem Vermittlungsrechner des Centrums für Informations- und Sprachverarbeitung ausgeführt. Für alle weiteren Schritte empfiehlt sich zur Überprüfung der Zwischenergebnisse das Arbeiten auf einem lokalen Gerät. Sind Administratorrechte vorhanden, können die erstellten Werkzeuge einfach mit einem Installationsskript systemweit installiert werden.[86] Ebenso ist ein lokaler Aufruf möglich, sofern die benötigten Softwarepakete wie der Python-Interpreter vorhanden sind.[87] Bis zur Ausgabe der fertigen Koordinatenvorschläge ist die getrennte Ausführung von acht Befehlen erforderlich. Sie sind als Bash-Skript implementiert, mit Parametern weitestgehend voreingestellt und rufen im Hintergrund die mit Python erstellten Werkzeuge auf. Ein Initialisierungsprogramm übernimmt nach erfolgter Verkleinerung das Anlegen eines nach dem Dokument benannten Arbeitsordners, das Herunterladen der Bilddateien vom Vermittlungsrechner, das Einkopieren der Editionsdaten sowie die Erstellung einer Konfigurationsdatei. Letztere trägt den Namen „Konfiguration.txt“ und hält Parameter für den weiteren Ablauf bereit.[88] Die Initialisierungsanweisung ist über den Befehl „Initialisierung“ mit dem Dokumentnamen als Parameter aufzurufen.[89]

4.2. Preprocessing – Vorbearbeitung

4.2.1. Bildverkleinerung

Die Bilder wurden vom Wittgensteinarchiv auf einem Vermittlungsrechner des CIS abgelegt. Für die Digitalisierung empfiehlt die DFG in ihren Praxisregeln eine Auflösung von 300 DPI.[90] Die

86 Alle entwickelten Programme und Skripte finden sich auf der CD-Rom und im GitLab Repositorium coordinates-extractor, Netz: https://gitlab.cis.uni-muenchen.de/CAST/coordinates-extractor, Stand: 22.08.2019. Aufruf: „bash Installation.sh“ Mit „bash Deinstallation.sh“ können die Skripte wieder aus „/usr/local/bin“ entfernt werden. Die Installation ist für Debian-basierte Systeme ausgelegt. Ausnahme: „Bildverkleinerung.sh“, vgl. hierzu Anm. 93.

87 Die benötigten Zusatzpakete werden automatisch installiert. Die Python-Module sind in der Datei requirements.txt aufgelistet.

88 Für die Benutzung sind entprechende Berechtigungen auf der CIS-Infrastruktur erforderlich.

89 Beispielaufruf: „Initialisierung Ms-101“.

90 Deutsche Forschungsgemeinschaft (Hg.), DFG-Praxisregeln „Digitalisierung“, Bonn, 2016, S. 15f., Günter Mühlberger, Die automatisierte Volltexterkennung, Marburg, 2015, S. 98f.

Dateien zum Nachlass liegen im TIF-Format mit deutlich größerer Auflösung vor. Sie sind in einem ersten Schritt zu verkleinern, um in Bezug auf Rechenzeit, Speicherverbrauch und Ausgabeergebnis eine effiziente Bearbeitung sicherzustellen. Höhere Auflösungen garantieren keineswegs bessere Ausgaben. Bei zu großen Bildern kann die Qualität der Ergebnisse sogar wieder abnehmen.[91]
Eine gute Möglichkeit zur kommandozeilenbasierten Stapelverarbeitung bieten die Befehle „convert“ und „mogrify“ aus dem quelloffenen ImageMagick-Paket.[92] Sie erlauben umfassende Feineinstellungen mittels Parametern. Die Konfiguration nach einem Fachartikel zum Thema hat die überzeugendsten Ergebnisse hervorgebracht.[93] Die Quellbilder sind bis zu zehn Megabyte groß. Die Ausgabebilder haben nach erfolgter Verkleinerung dagegen nur noch eine Größe von circa einem Megabyte. Für sie wurde ein eigenes Unterverzeichnis „high_density_jpg“ erzeugt. Eine Vereinheitlichung der Bildbreite auf 2500 Pixel erleichtert die Randentfernung. Das Initialisierungsskript kopiert nach erfolgter Bildverkleinerung automatisch die verkleinerten Bilder für alle weiteren Schritte auf dem lokalen Rechner.

4.2.2. Randentfernung

Die abfotografierten schwarzen Ränder würden die Binarisierung und Texterkennung stören und müssen daher entfernt werden. ScanTailor wäre in der Lage, diese Aufgabe zu übernehmen, verschiebt aber mit der Bildung einheitlicher Rahmen den Textbereich. Die ursprünglichen Bildkoordinaten auf den Seiten würden deshalb nicht mehr zu den Quellbildern als Darstellungsziel in WiTTFind passen. Weder über die grafische, noch die kommandozeilenbasierte Schnittstelle können Einzelschritte getrennt voneinander aufgerufen werden. Wie Abb. eins zu entnehmen entfernt die Autoerkennung immer wieder relevante handschriftliche Ergänzungen aus dem Textbereich.
Die Bilder wurden von professionellen Dienstleistern erstellt, liegen in hoher Qualität vor und bilden für Zwecke der Texterkennung einen sehr guten Ausgangspunkt. Die Position der Dokumente auf der Auflagefläche des Einlesegerätes wurde beim Umblättern nicht verändert. Außerdem sind die Seiten bereits genau ausgerichtet und störungsfrei abfotografiert. Abgesehen von der Entfernung der schwarzen Ränder und einer Binarisierung sind keine weiteren Vorbearbeitungsmaßnahmen zwingend erforderlich. Von einer Verwendung von ScanTailor wurde

91 Erfahrungsaustausch auf verschiedenen Konferenz, vgl. hierzu: [o.A.], Optimal image resolution (dpi/ppi) for Tesseract 4.0.0 and eng.traineddata?, Netz: https://groups.google.com/forum/#!msg/tesseract-ocr/Wdh_JJwnw94/24JHDYQbBQAJ, Stand: 10.08.2019, Mühlberger, Die automatisierte Volltexterkennung, Marburg, 2015, S.98f.
92 ImageMagick, Netzauftritt: https://imagemagick.org/script/convert.php Stand: 10.08.2019.
93 Dave Newton, Efficient Image Resizing With ImageMagick, [o. O.], 2015, Netz: https://www.smashingmagazine.com/2015/06/efficient-image-resizing-with-imagemagick/, Stand: 10.08.2019, Beachte hierzu die Anleitung auf der beiliegenden CD-Rom bzw. im GitLab-Repositorium: https://gitlab.cis.uni-muenchen.de/wast/wittgenstein-faksimile-download/blob/Bildoptimierung-issue-945/Vorschlag_Bildverkleinerung_Landes/Anleitung_zur_Bildverkleinerung.txt, Stand: 10.08.2019.

deshalb abgesehen und auch in diesem Fall auf ImageMagick zurückgegriffen. Die Faksimile sind mit einem geringen Überstand bereits in einzelne Seiten getrennt nach einem weitgehend einheitlichen Schema abgespeichert. Auf den Dokumentnamen folgt durch ein Komma getrennt die Seitennummer. Direkt angehängt ist die Ausrichtung einzelner Blätter nach recto bzw. verso, repräsentiert durch die Buchstaben „a“ oder „b“ bzw. „r“ oder „v“. Je nach vorliegendem Fall befindet sich der Seitenbereich des Bildes in einem bestimmten Rechteck. Es sind sieben verschiedene „Bildklassen“ zu unterscheiden. So existieren einerseits Aufnahmen von arabisch nummerierten geraden und ungeraden Blättern jeweils in einer Recto- und Versoversion. Gebundene Dokumente wie Notizbücher kommen ohne Recto- oder Versoangabe aus. Bei Vor- und Nachlaufseiten, dem Buchdeckel, wie auch Rücken und Boden wurden die Seitenbilder römisch nummeriert. Sie sind wiederum je nach Dokument teilweise in einer Recto- und Verso-Variante vorhanden. Außerdem gibt es einige aus dem Schema völlig herausfallende Sonderbilder wie z. B. Doppelaufnahmen.[94]

Für das Lesen der Bildnamen ist ein Programm mit dem Namen Seitendurchlauf zuständig. Es iteriert über alle Dateien des „high-density-jpg“-Verzeichnisses und parst mit regulären Ausdrücken und Stringoperationen den Dateinamen.[95] In der dokumentspezifischen Konfigurationsdatei werden in sechs Variablen die für die verschiedenen Bildklassen notwendigen Pixelwerte vorgehalten. Auf den Dokumentnamen in der ersten Zeile folgen die einzelnen Pixelvariablen für die Randentfernung. Sie sind durch das Initialisierungsskript bereits erstellt und müssen vor Beginn manuell belegt werden. Dafür sind die Pixelwerte der Ränder oben, unten, links und rechts für die verschiedenen Bildklassen durch den Bearbeiter zu messen und in die Konfigurationsdatei einzutragen. Für die Ränder oben und unten gelten die gleichen Pixelwerte wie für alle Bilder. Die in der Konfiguration gespeicherten Informationen können von Bash-Skripten mit einer einfachen „source“-Anweisung und innerhalb der Python-Werkzeuge mit einer eigenen Importfunktion gelesen werden.[96] Für alle Bilder reagiert das Skript für den Seitendurchlauf entweder mit einem entsprechenden Aufruf zur Randentfernung oder mit einem Hinweis an den Nutzer auf Sonderfälle. Für letztere wird die Randentfernung mit den Parametern für geradzahlige Seiten aufgerufen, um sie in der Verarbeitungskette zu halten. Das Ergebnis muss für diese Seiten manuell geprüft und ggf. korrigiert bzw. Doppelaufnahmen entfernt werden.[97] Seitendurchlauf übergibt die gelesene Information an das eigentliche Skript zur Randentfernung.

94 Z.B. Ms-104, Ms-138, Ts-202.

95 Aufruf „Seitendurchlauf“ im Arbeitsordner. Achtung: Nur mit Übergabe des Parameters „1“ wird die Randentfernung aufgerufen. So kann das Parsen vor der Verkleinerung über die Konsolenausgabe geprüft werden.

96 Demo.py, Methode leseKonfigurationsdatei.

97 Das Entfernen von Doppelaufnahmen könnte automatisiert werden, bei manueller Entfernung kann der Bearbeiter das bessere Bild selbst auswählen.

Darin kommt erneut der convert-Befehl mit eigener Konfiguration zur Verwendung. Für die vorliegende Aufgabe ist die Funktion, einfache weiße Rechtecke über die fotografierten Buchränder zu legen, das Mittel der Wahl. Letzteres geschieht mittels dem Aufruf „-draw“ in Kombination mit der Option „rectangle“ unter Verwendung der in der Konfigurationsdatei hinterlegten Pixelwerte.[98] Die randlosen Dateien werden in das neue Verzeichnis „ohne_Rand_0“ abgelegt.

4.2.3. Binarisierung

Der vierte Schritt in der Verarbeitungskette ist die Binarisierung der Bilddokumente. Sie ist Grundvoraussetzung für eine effiziente Bearbeitung der Dateien mit einer Erkennungsanwendung. Zwar bieten verschiedene OCR-Programme eigene Binarisierungssysteme an. Um die Zwischenergebnisse manuell korrigieren zu können, eine bessere Vergleichbarkeit der Ergebnisse zu erzielen und den Verarbeitungsprozess einheitlicher zu gestalten wurde jedoch erneut auf die Werkzeuge von ImageMagick zurückgegriffen. Für die Binarisierung ist als zentraler Parameter ein prozentualer Schwellenwert an convert zu übergeben, der sich für die im Nachlass bearbeiteten Bilder je nach deren Helligkeit und der verwendeten Schreibmaterialien zwischen 60 und 85 Prozent bewegte. Auch für diesen vierten Verarbeitungsschritt wurde ein Skript erstellt. Es akzeptiert als Parameter den prozentualen Schwellenwert.[99] Standardmäßig sind 60 Prozent voreingestellt. Das Programm durchläuft alle im Quellverzeichnis befindlichen Daten und speichert das Endergebnis in einem nach dem Schwellenwert benannten Unterverzeichnis ab.

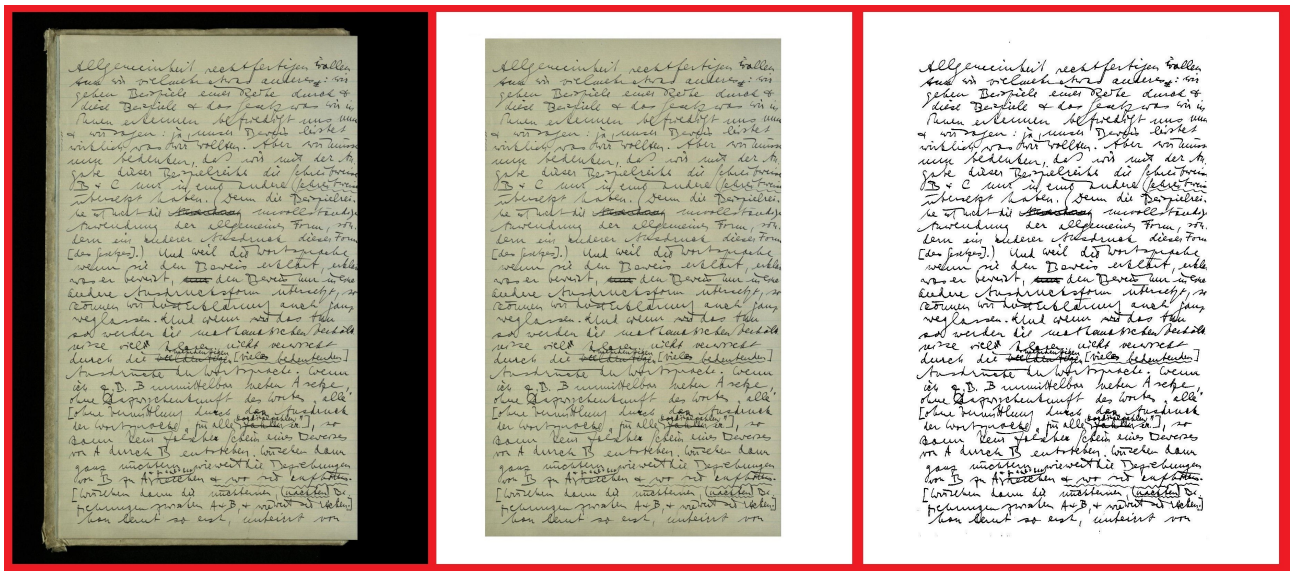


Abb. 8: Vorbearbeitung, von links nach rechts am Beispiel Ms-114,2v

98 Skript Randentfernung, Zeilen 84-87.
99 Aufruf „Binarisierung 75“ - Binarisierung mit einer Schwelle von 75%. Ohne Parameter sind 60% voreingestellt.

4.3. Texterkennung

Die Bilder liegen nun bereit zur Verarbeitung durch verschiedene Werkzeuge zur Text- bzw. Zeichenerkennung. Für den weitaus größten Teil des bearbeiteten Bildmaterials wurde Tesseract verwendet. Weil Tesseract vier in verschiedenen Alpha- und Beta-Versionen ohne erkennbare Gründe immer wieder defekte oder leere HOCR-Ausgaben erzeugte, wurde für die Stapelverarbeitung der Einsatz der stabilen Version 3.04.01 der Vorzug gegeben.

Die automatisierte Bearbeitung aller Bilder mit den gewünschten Parametern kapselt das Programm „Texterkennung“. Es genügt der gleichnamige Aufruf auf der Kommandozeile als fünfter Schritt in der Arbeitskette. Wegen seiner einfachen Handhabung und nativen Unterstützung wurde als Ausgabeformat HOCR voreingestellt. Deutsch ist als überwiegend benutzte Sprache ausgewählt, Englisch oder eine anderes Modell sind über einen Parameter konfigurierbar.[100]

```
for bild in $bilder

do
    echo '################################################################'
    echo "Texterkennung :" $bild
    echo '----------------------------------------------------------------'

    breite=`identify -format "%w" $bild`
    hoehe=`identify -format "%h" $bild`
    echo Bildgröße:
    echo Breite: $breite
    echo Höhe: $hoehe

    rohname=${bild%.jpg}


    tesseract $bild zres_$rohname -l $lang -psm $psm hocr
    echo '----------------------------------------------------------------'
```

Abb. 9: Schleife zur automatisierten Texterkennung

Tesseract wurde herangezogen, weil es mit seinem robusten Universalmodellen in der überwiegenden Zahl der Fälle die besten Endergebnisse hervorgebracht hat. Der Datenbestand im Bereich der Typoskripte fällt relativ heterogen aus. So wurden die Dokumente mit verschiedenen Schreibmaschinen geschrieben. Je nach Abnutzung des Farbbandes und des verwendeten Papiers variieren die einzelnen Zeichen zum Teil stark. Gerade bei Durchschlägen auf säurehaltigem Papier, bei Vergilbungen oder bei schwachem Farbband sind die Konturen vieler Buchstaben unscharf oder verwischt.[101] Es kommen ganze handschriftliche Seiten,[102] Ergänzungen wie Einfügungen,[103] Streichungen[104] oder Kommentare hinzu, die die Schaffung eines fundierten GroundTruth Bestandes

100 Aufruf: „Texterkennung 60 3 eng“, für Englisch. Erster Parameter Binärschwelle, zweiter Segmentierungsmodus.
101 Ts-202, 203.
102 Z.B.: Ts-204,10ar.
103 Z.B.: Ts-204,42ar.
104 Z.B.: Ts-204,11r .

weitestgehend uninteressant machen.[105] Der Aufwand eines spezifischen Trainings auf den in der Regel nicht mehr als 150 Seiten umfassenden Dokumenten stünde in keinem Verhältnis mit dem erzielten Nutzen zumal eine fehlerfreie Transkription bereits vorhanden ist. Die Erkennung mit dem von OCR4all bereitgestellten Antiqua Universalmodellen[106] war der Ausgabe von Tesseract überwiegend unterlegen. Auch beim direkten Vergleich der Ergebnisse von Tesseract und Ocropus zeigte sich das Universalmodell von Tesseract im Bereich der Typoskripte als besser geeignet. Die Ausgabe ist zwar bei weitem nicht fehlerfrei, für eine Alignierung sind die erkannten Wörter von Tesseract jedoch in den meisten Fällen gut geeignet. Die Ergebnisse im Falle von Manuskripten waren mit allen quelloffenen OCR-Anwendungen vollkommen unbrauchbar. Für die Alignierung von Text und Bild als nächstem Arbeitsschritt stellte sich das Lesen von HOCR- wie den XML-Editionsdaten als weitere Aufgabe.[107]

4.4. Ermittlung von Koordinatenvorschlägen

4.4.1. Zwei Datenbestände: Edition und OCR

Zur Erzeugung von Koordinatenvorschlägen kommt die Edition als zweite Informationsquelle ins Spiel. Sie stellt praktisch das fertige Gegenstück zu den OCR-Daten dar. Aus dem Wittgensteinarchiv stehen mehrere XML-basierte Dokumente bereit. Die Dateien entsprechen den Vorgaben der Text Encoding Initiative (TEI).[108] Das TEI-Format bietet zur Speicherung von digitalen Editionen einen großen Freiraum. So können beispielsweise verschiedene Strukturierungen ausgewählt oder eigene Elemente definiert werden. Von Edition zu Edition fallen TEI-Daten trotz genauer Festschreibungen teilweise sehr unterschiedlich aus. Im vorliegenden Fall lag eine diplomatische Editionsdatei mit umfangreichen Informationen aller Art sowie eine normalisierte Ausgabe vor. Diese Norm-XML-Daten repräsentieren aus der Sicht der Editoren den letzten Willen des Autors. Dazu korrespondierend existiert eine im Browser darstellbare passende HTML-Datei im GIT-Repositorium.[109]

Für die Verknüpfung von digitaler Edition und Faksimile hat sich aufgrund der geringeren Informationsdichte die Verwendung der weniger detaillierten Norm-XML Dateien als vorteilhaft herausgestellt. Elemente wie verschiedene Lesarten, die eine Alignierung stören könnten, sind bereits vorab entfernt. Alle Wörter und Zeilen wie auf der Seite vorhanden werden abgesehen von orthografischen Korrekturen vollständig im XML-Text repräsentiert.

105 Z.B.: Ts-204,16r.
106 Calamari-OCR, GitHub-Repositorium, Netz: 10.08.2019.https://github.com/Calamari-OCR/calamari_models, Stand: 10.08.2019.
107 Die HOCR-Dateien werden in einem eigenen Unterverzeichnis abgelegt.
108 Text Encoding Initiative (TEI), Netzauftritt: https://tei-c.org/, Stand: 10.08.2019.
109 witt-data, GitLab-Repositorium, Netz: https://gitlab.cis.uni-muenchen.de/wast/witt-data, Stand: 10.08.2019.

Zentrale Struktureinheiten der Editionsdaten sind AB-Elemente. Sie durchziehen alle Dokumente des Wittgensteinarchivs und speichern eindeutige Siglen als Schlüsselattribute. Die Siglen setzen sich zusammen aus dem Dokumentnamen, gefolgt von einem Komma und der Seitennummer. Eine Bemerkungsnummer folgt in eckigen Klammern.[110]

```
<ab n="Ts-235,1[1]" ana="abnr:1">
   <seg type="contents">
      <s n="Ts-235,1[1]_1" ana="facs:Ts-235,1 abnr:1 satznr:1">
      ‘Sprache’, ‘Satz’, ‘Wort’, Begriffe des Alltags.</s>
   </seg>
</ab>
```

Abb. 10: Anonymer Block „Ts-235,1[1]“ aus der Edition

Für Bemerkungen, die sich über mehrere Seiten erstrecken, sind Mehrfachsiglen vorhanden. Sie werden durch das Schlüsselwort „et“ eingeleitet. Es folgt die Seitennummer ohne Dokumentname sowie die weiteren Bemerkungsnummern in eckigen Klammern.[111]

4.4.2. Parsen und strukturieren von Editions- und HOCR-Daten

Zum Parsen der XML-Daten wurde die Python-Bibliothek LXML[112] herangezogen. LXML bietet im Vergleich zu XML-ETREE eine erweiterte Funktionalität. So ist beispielsweise nicht nur der gesamte Text innerhalb eines Elements ansprechbar, sondern auch der Folgetext als Tail-Attribut referenziert. Der Nutzer hat zahlreiche Methoden zur Verfügung, die das Iterieren und Traversieren über die XML-Baumstruktur erlauben. Zur Alignierung der beiden Datenbestände war die Schaffung einer eigenen Python-Objektstruktur von Nöten. Die Edition gab zunächst das Speichern von anonymen Blöcken vor. Wünschenswert erschien jedoch der zeilenweise Vergleich pro Seite. Inkompatibel mit der angestrebten Repräsentation sind die seitenübergreifenden anonymen Blöcke aus der Edition mit ihren Mehrfachsiglen. Es musste daher aus der gelesenen Editionsdatenstruktur eine Repräsentation nach tatsächlichen Textseiten geschaffen werden. Die vorhandenen Editionsobjekte waren entsprechend umzustrukturieren. Mehrfachsiglen wurden getrennt, um auf die einzelnen Bemerkungen zugreifen und die einzelnen Seiten speichern zu können. Es waren neue Seitenobjekte notwendig,[113] in denen sich Informationen aus Edition und OCR ablegen und algorithmisch miteinander verknüpfen lassen. Als Gegenstück zu den Editionszeilen kamen Felder für die OCR-Zeilen als auch für alignierte Zeilen hinzu.

110 Beispiel: Ms-101,1r[1].
111 Beispiel: Ms-101,1r[2]et2r[1].
112 LXML, Netzauftritt: https://lxml.de/, Stand: 10.08.2019.
113 Vgl. hierzu im Quelltext die Klasse: „alignedPage“ in der Datei Aligned_Result.py.

4.4.3. Erzeugen von Koordinatenvorschlägen für Typoskripte

Nach dem Parsen der Edition werden die HOCR-Daten ebenso mit LXML eingelesen und die Koordinateninformation in die erzeugten Seitenobjekte geschrieben.

Die Zeilen haben in beiden Datenbeständen eine vorgegebene Sortierung von oben nach unten. Sofern sie durch das OCR-System richtig erkannt wurden, hat jede Zeile im Idealfall je ein entsprechendes Gegenstück in den Editionsdaten.

Vorausgesetzt einer korrekten Transkription mit Zeilenumbrüchen können die einzelnen Zeilen, Wörter oder Buchstaben miteinander verglichen und ein gewichtetes Ähnlichkeitsmaß aufgestellt werden. Zu jeder Editionszeile wird dafür eine Indexliste erstellt, die die maximale Ähnlichkeit der vorhandenen OCR-Zeilen zur zugehörigen Editionszeile repräsentiert. Je höher der Zahlenwert an einer Postion desto größer die Ähnlichkeit zwischen der Editionszeile und der OCR-Zeile am entsprechenden Index.

Auf die Indexliste sind verschiedene Gewichtungsmaße wie zum Beispiel die gleiche Anzahl an Wörtern in beiden Zeilen, ein Übereinstimmen der Vorgänger- und Nachfolgezeilennummer oder Gesamtbuchstabenlänge entsprechend aufrechenbar. Ebenso möglich zu berücksichtigen ist ein Auftreten von Wörtern aus der Edition in der OCR-Zeile. „Pluspunkte“ können auch vergeben werden, wenn die Reihenfolge der einzelnen Wörter übereinstimmt.

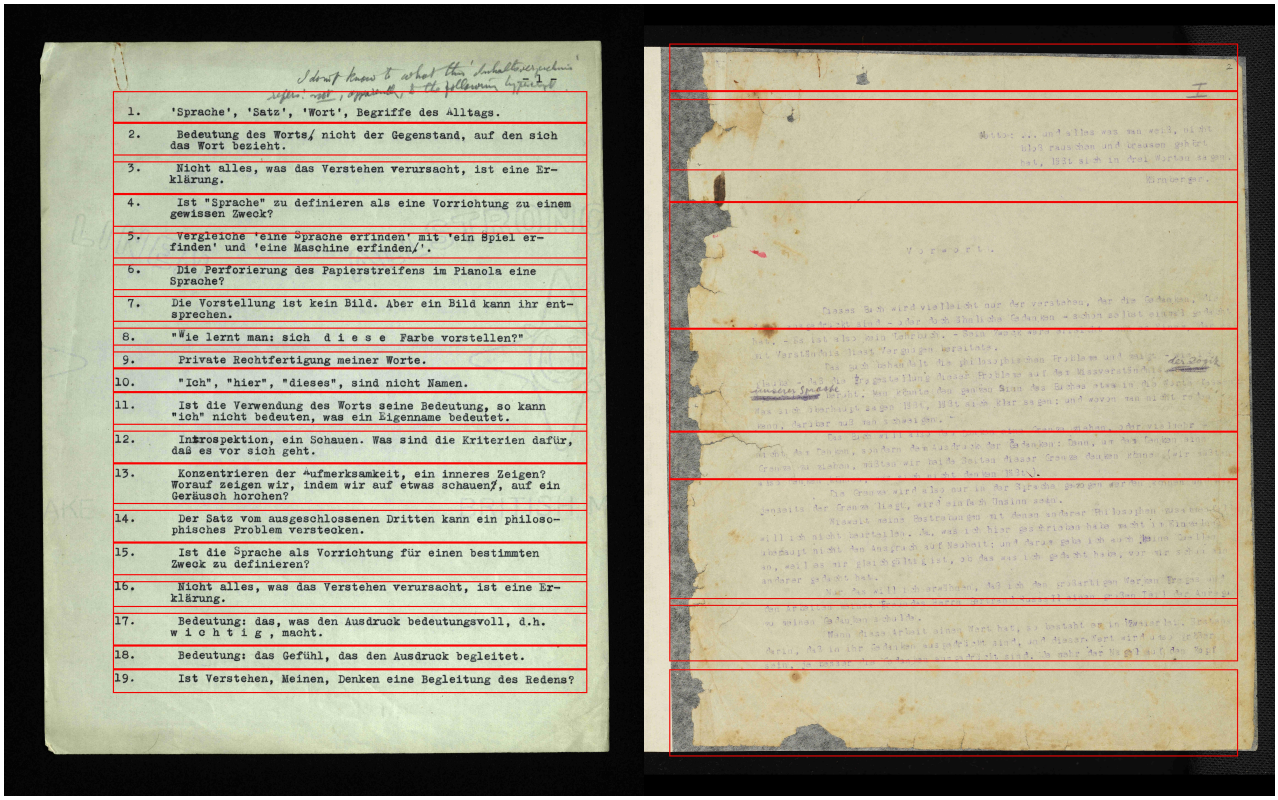
```
def allign_lines(edition_lines, ocr_lines, page_id):
    alligned_page = Aligned_Page()
    alligned_page.id=page_id
    max_similarity_index_dict = OrderedDict()
    max_ocr_line_index = len(ocr_lines)
    for edition_line in edition_lines: # Iteration über die Editionszeilen
        compare_list=zero_list_builder(len(ocr_lines))
        splitted_edition_line = edition_line.text.split()
        [...]
        for ocr_line in ocr_lines: # Iteration über alle OCR-Zeilen
            splitted_ocr_line=ocr_line.text.split()
            if len(splitted_ocr_line) == len(splitted_edition_line):
                compare_list[counter] += 1 # Aufrechnen von Wertungspunkten
        [...]
    remove_multiple_indixes(max_similarity_index_dict) # Entfernen doppelter Indizes
    [...]
    return alligned_page
```

Abb. 11: Verkürzter Auszug aus dem Python-Quelltext zur Alignierung von Typoskripten

Ein häufiger Fehler in OCR-Ergebnissen ist die Falscherkennung einzelner Zeichen. Zwei Wörter lassen sich mit Hilfe des Levenshteinabstandes miteinander vergleichen.[114] Er definiert ein Ähnlichkeitsmaß auf der Basis der Zählung von Änderungsoperationen. Jede Operation wie ein Einfügen, Löschen oder Ändern wird gerechnet. Zwei identische Wörter haben somit den Levenshteinabstand von Null. Je mehr Änderungsoperationen erforderlich sind, desto größer wird

114 Levenshtein-Algorithmus, Netzauftritt: http://www.levenshtein.de/, Stand: 10.08.2019.

der Levenshteinabstand, desto unähnlicher sind die verglichenen Wörter. Bleibt das durchschnittliche Ergebnis für die Wörter einer Zeile unter einer festgelegten Grenze, so können dafür Pluspunkte vergeben werden.



1. 'Sprache', 'Satz', 'Wort', Begriffe des Alltags.
2. Bedeutung des Worts, nicht der Gegenstand, auf den sich das Wort bezieht.
3. Nicht alles, was das Verstehen verursacht, ist eine Erklärung.
4. Ist "Sprache" zu definieren als eine Vorrichtung zu einem gewissen Zweck?
5. Vergleiche 'eine Sprache erfinden' mit 'ein Spiel erfinden' und 'eine Maschine erfinden'.
6. Die Perforierung des Papierstreifens im Pianola eine Sprache?
7. Die Vorstellung ist kein Bild. Aber ein Bild kann ihr entsprechen.
8. "Wie lernt man: sich d i e s e Farbe vorstellen?"
9. Private Rechtfertigung meiner Worte.
10. "Ich", "hier", "dieses", sind nicht Namen.
11. Ist die Verwendung des Worts seine Bedeutung, so kann "ich" nicht bedeuten, was ein Eigenname bedeutet.
12. Introspektion, ein Schauen. Was sind die Kriterien dafür, daß es vor sich geht.
13. Konzentrieren der Aufmerksamkeit, ein inneres Zeigen? Worauf zeigen wir, indem wir auf etwas schauen?, auf ein Geräusch horchen?
14. Der Satz vom ausgeschlossenen Dritten kann ein philosophisches Problem verstecken.
15. Ist die Sprache als Vorrichtung für einen bestimmten Zweck zu definieren?
16. Nicht alles, was das Verstehen verursacht, ist eine Erklärung.
17. Bedeutung: das, was den Ausdruck bedeutungsvoll, d.h. w i c h t i g , macht.
18. Bedeutung: das Gefühl, das den Ausdruck begleitet.
19. Ist Verstehen, Meinen, Denken eine Begleitung des Redens?

Abb. 12: Einfaches Beispiel mit guter Alignierung (Ts-235,1) vs. problematischer Fall (Ts-202,Ir)

Als ein häufiger Störfaktor haben sich fehlende[115] oder fehlerhafte[116] Zeilen- und Seitenumbrüche in der Edition und Fehlsegmentierungen bei der OCR herausgestellt. Mittels Parameterübergabe wie zur Begrenzung der Spaltenzahl konnte diesen Fehlern zumindest teilweise begegnet werden. Weitaus problematischer gestalteten sich falsch erkannte handschriftliche Ergänzungen oder Streichungen, die es zu entfernen galt. Dafür wurden verschiedene Filterfunktionen implementiert, die mit Stringoperationen und regulären Ausdrücken versuchen, fehlerhafte Ausgaben zu erkennen. Beispielsweise können OCR-Zeilen gelöscht werden, die eine auffällig große Zahl an Sonderzeichen wie „@“, „$“oder „§“ enthalten, die leer oder zu kurz sind oder in denen keine Zeichen aus dem Alphabet vorkommen. Die Gewichtungs- und Filtermethoden sind je nach Datenbestand und Fehlermuster im Dokument individuell wiederhol-, hinzuschalt- oder abwählbar.[117] Dabei war die experimentelle Kombination verschiedener Wertungssysteme bei der Zusammenstellung unumgänglich. Im Quelltext findet sich eine relativ stabile Auswahl, die für die meisten Dokumente gute Resultate ergab.

115 Z.B. Ms-114,112r[1].
116 Z.B. Ts-220,48 Zeile 1.
117 Dafür müssen Änderungen in der Methode Allign Lignes in Alignment.py (Abb. 10) durchgeführt werden.

Das beschriebene Verfahren kann als sechster Schritt der Arbeitskette mit dem Befehl „Koordinatenbestimmung T“ angestoßen werden. Für Typoskripte ohne Falscherkennungen wären bei fehlerfreien Editionsdaten Ergebnisse erzielbar, die eine menschliche Nachkorrektur theoretisch weitestgehend überflüssig machen. In der Praxis jedoch waren weder die OCR-Daten noch die Editionsdaten frei von Fehlern.

4.4.4. Erzeugen von Koordinatenvorschlägen für Manuskripte

4.4.4.1. Bestimmung des Textbereiches

Wie bereits angesprochen war die Text-Ausgabe aller OCR-Anwendungen bei handgeschriebenen Dokumenten völlig unbrauchbar. Sie machen jedoch einen Großteil des Gesamtbestandes aus. Auf Grundlage der HOCR-Daten ist es trotzdem möglich, den ermittelten Textbereich auf der Seite aus den Bounding-Box Attributen auszulesen. Er wird von Tesseract weitgehend korrekt erkannt. Die Koordinatenwerte erlaubten es, zusammen mit den Informationen aus der Edition, Koordinatenvorschläge zu berechnen.

Beim Parsen der HOCR-Daten wurden die erforderlichen Informationen ausgelesen. Auf einen Zusatzbefehl hin ist eine Schlüssel-Wert-Struktur im JSON-Format exportierbar.[118] Zur Kontrolle wird in einem Unterverzeichnis der erkannte Textbereich in die Faksimile gezeichnet. Das zugehörige Programm kann als sechster Schritt mit dem Aufruf „Hervorhebung“ gestartet werden. Für das entsprechende Dokument wird im Arbeitsverzeichnis eine Datei „<Dokumentname>.conf.json“ angelegt. Darin sind pro Seite mit dem Dateinamen als Schlüssel vier Koordinatenwerte, die den ermittelten Textbereich abgrenzen, abgespeichert. Außerdem wird die Breite und Höhe des Textbereiches hinterlegt. Diese Informationen bilden die Grundlage zum Abschätzen der Koordinaten.

4.4.4.2. Abschätzen von Koordinatenvorschlägen

In einem ersten Ansatz wurde die Höhe des gesamten Textbereiches einer Seite durch die Zahl der Siglen dividiert und in gleich große Unterabschnitte aufgeteilt. Bei diesem Vorgehen bleibt der unterschiedliche Umfang der Bemerkungen unberücksichtigt. Die Schätzergebnisse waren nur bei drei gleich langen Textblöcken auf der Seite annähernd passend. Der Nachkorrekturaufwand war bei diesem Vorgehen entsprechend groß.

118 Aufruf: „Hervorhebung“.

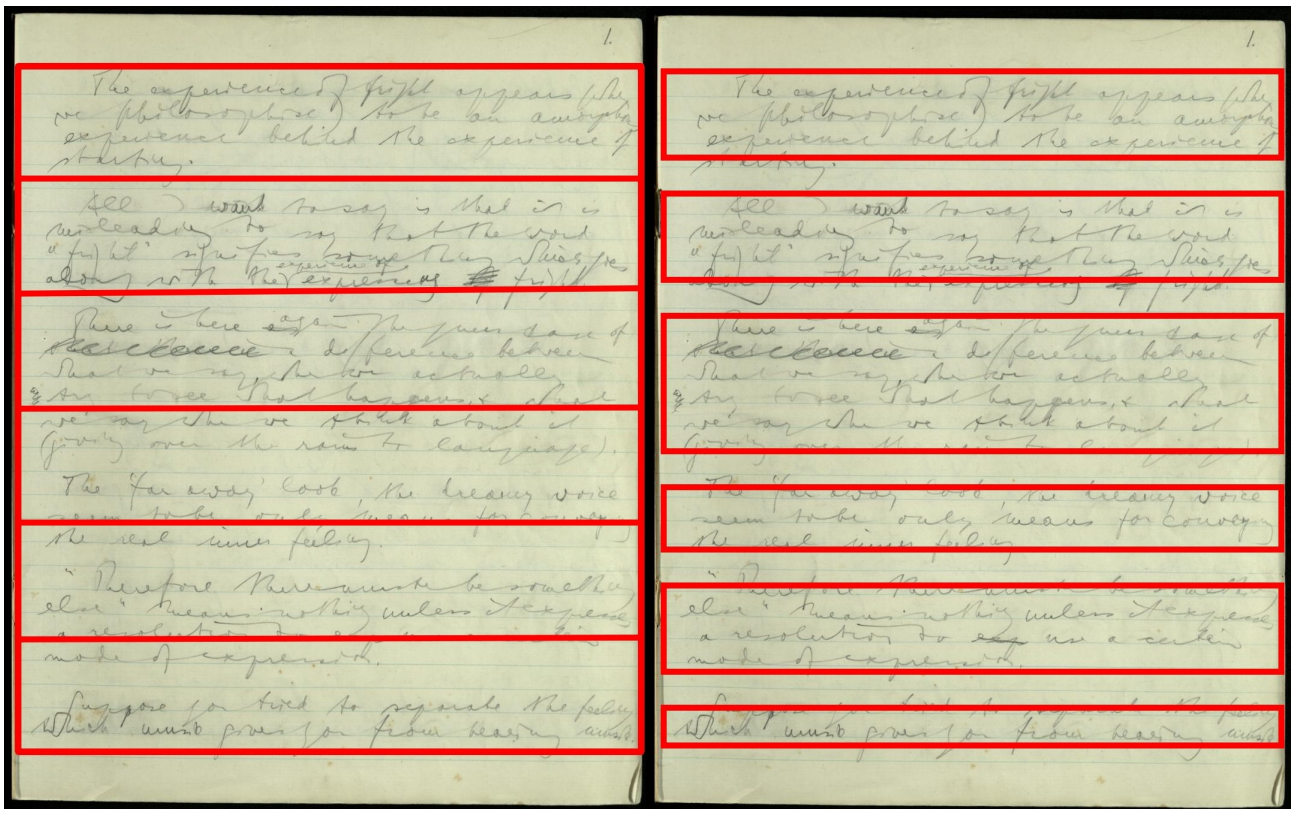


Abb. 13: Bildung gleich großer Blöcke vs. zeilenbasierte Schätzung (Ms-148,1r)

In einem zweiten Ansatz wurde deshalb versucht, den Textbereich in gleichgroße Zeilen aufzuteilen und mit Hilfe der Zeilenzahl die Textblöcke entsprechend abzugrenzen. Dafür wurden die Anzahl und durchschnittliche Höhe der Zeilen auf der Seite durch eine Zählung der linebreak- oder Zeilenumbruchelemente in den Editionsdaten bestimmt. Zwischen den Blöcken wurde eine Leerzeile aufgerechnet. Mit diesem Ansatz war eine deutliche Verbesserung erreichbar. Der Textbereich wurde nun annähernd proportional zur Edition abgetrennt.

Als problematisch stellte sich jedoch heraus, dass in den Editionsdaten manche Zeilenumbrüche fehlten oder dass die Schriftgröße und folglich die Zeilenhöhe sich unterschieden. Leerzeilen zwischen Absätzen waren meist stark unterschiedlich ausgeprägt, die Einführung eines Zeilenmaßes nur für wenige Dokumente möglich.

Auch die Verwendung von verschiedenen algorithmischen Zeilensegmentierungswerkzeugen wie dem gpageseg-Modul von Ocropus oder Kraken-Segment lieferte nur ungenügende bis bestenfalls ausreichende Ergebnisse.

Die Feineinstellung mittels Parametern erbrachte keine zufriedenstellende Verbesserung der Ausgaben. Oftmals gehen Zeilen ineinander über und sind nicht ohne Überschneidungen abgrenzbar. Auf manchen Seiten wurden zumindest einzelne Zeilen oder Textblöcke annähernd richtig erkannt. In vielen Fällen gelang dies nicht einmal in Bezug auf die Schreibrichtung. Ähnlich

wie die Ausgabe der Zeichenerkennung waren die Ergebnisse der automatischen Seitensegmentierung praktisch unverwertbar.

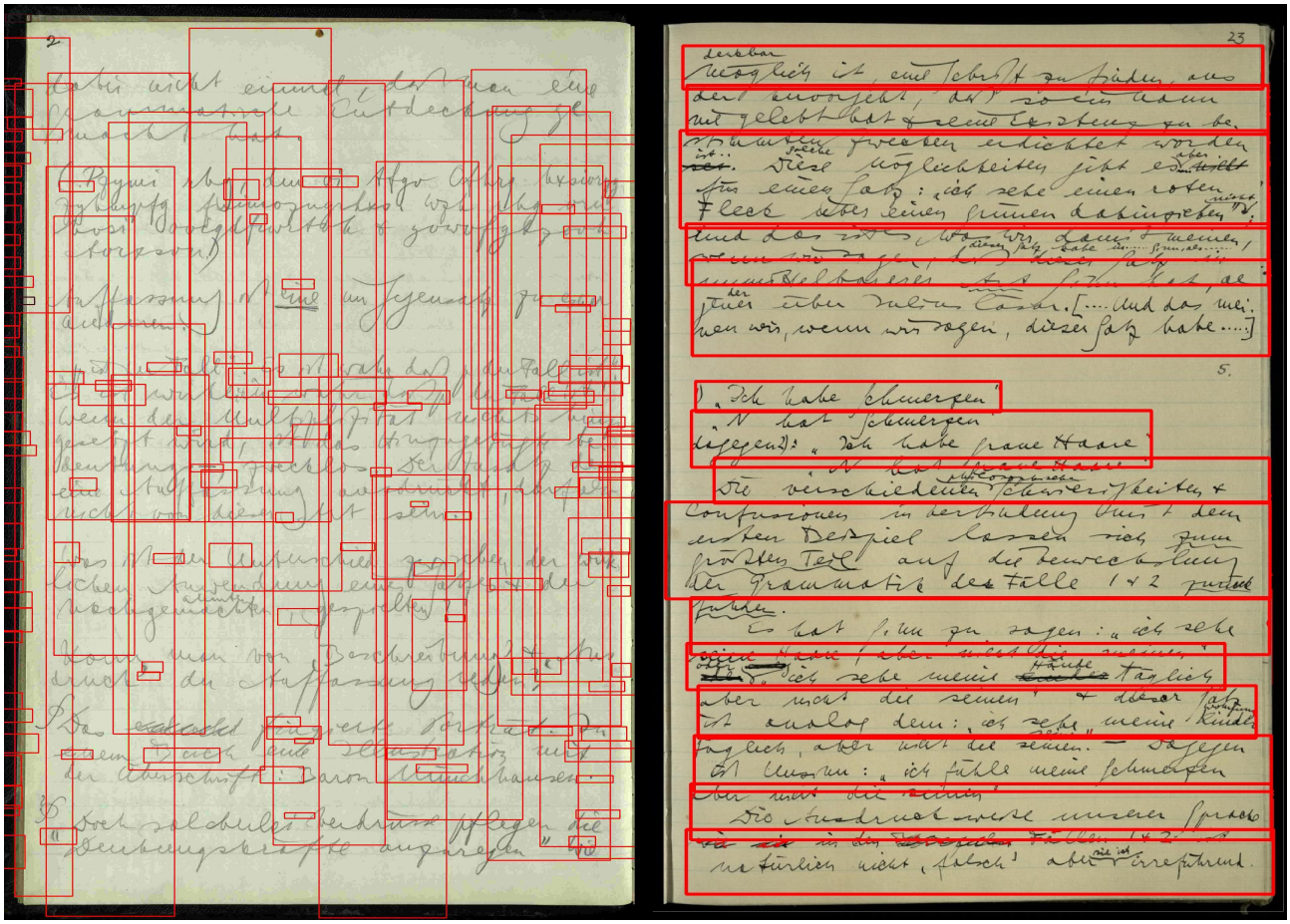


Abb. 14: Unbrauchbare (Ms-111,2) und verwertbare Segmentierungsergebnisse (Ms-114,23)

Zwar konnte die Qualität der Ergebnisse mit dem zeilenbasierten Ansatz pro Paket gesteigert werden, ein praktikables Verfahren wurde jedoch erst in einem dritten Ansatz erreicht.

```
def calculate_text_percentages_and_coordinates_of_export_blocks(self):
        [...]
        # Die Zeichenzahl der gesammten Seite wird auf 100 Prozent gesetzt:
        hundred_percent=len(self.text)
        # Ein Prozent wird berechnet:
        one_percent=hundred_percent/100
        [...]
        for export_block in self.export_blocks:
            export_block.percentage_of_textarea = len(export_block.text)/one_percent
            # Die Höhe des aktuellen Blocks wird berechnet:
            height_of_recent_block=one_percent_of_height*export_block.percentage_of_textarea
            [...]
            # Leeres Koordinatenfeld im Format x0 y0 x1 y1 wird angelegt und gefüllt:
            coordinates = [0, 0, 0, 0]
            coordinates[0] = self.text_area['MARGIN_LEFT'] # x0
            coordinates[2] = self.text_area['MARGIN_RIGHT'] # x1
            coordinates[1] = recent_upper_edge_coordinate #y0
            coordinates[3] = recent_upper_edge_coordinate + height_of_recent_block # y1
            export_block.coordinates = coordinates
            # Die Oberkannte wird aktualisiert:
            recent_upper_edge_coordinate=recent_upper_edge_coordinate+height_of_recent_block
```

Abb. 15: Verkürzte Methode zur zeichenbasierten Koordinatenschätzung

In jeder Editionsdatei ist die Anzahl der Zeichen auf einer Seite dank der Siglen als Absatz- und Seitentrenner ermittelbar. Über eine einfache Dreisatzrechnung kann der prozentuale Anteil des einzelnen Absatzes an der Gesamtzeichenzahl ermittelt werden. Dieser ist auf einen entsprechend hohen Teilabschnitt des Textbereiches übertragbar. Somit lassen sich relativ präzise Bildausschnitte abschätzen.

Natürlich hängen die Ergebnisse von der korrekten Erkennung der Textbereiche auf der Seite ab. In den meisten Fällen gelingt es Tesseract, diese Information richtig zu ermitteln. Folgende Bildschirmfotos geben einen Eindruck von einer verhältnismäßig guten wie auch einer relativ schlechten Schätzung.

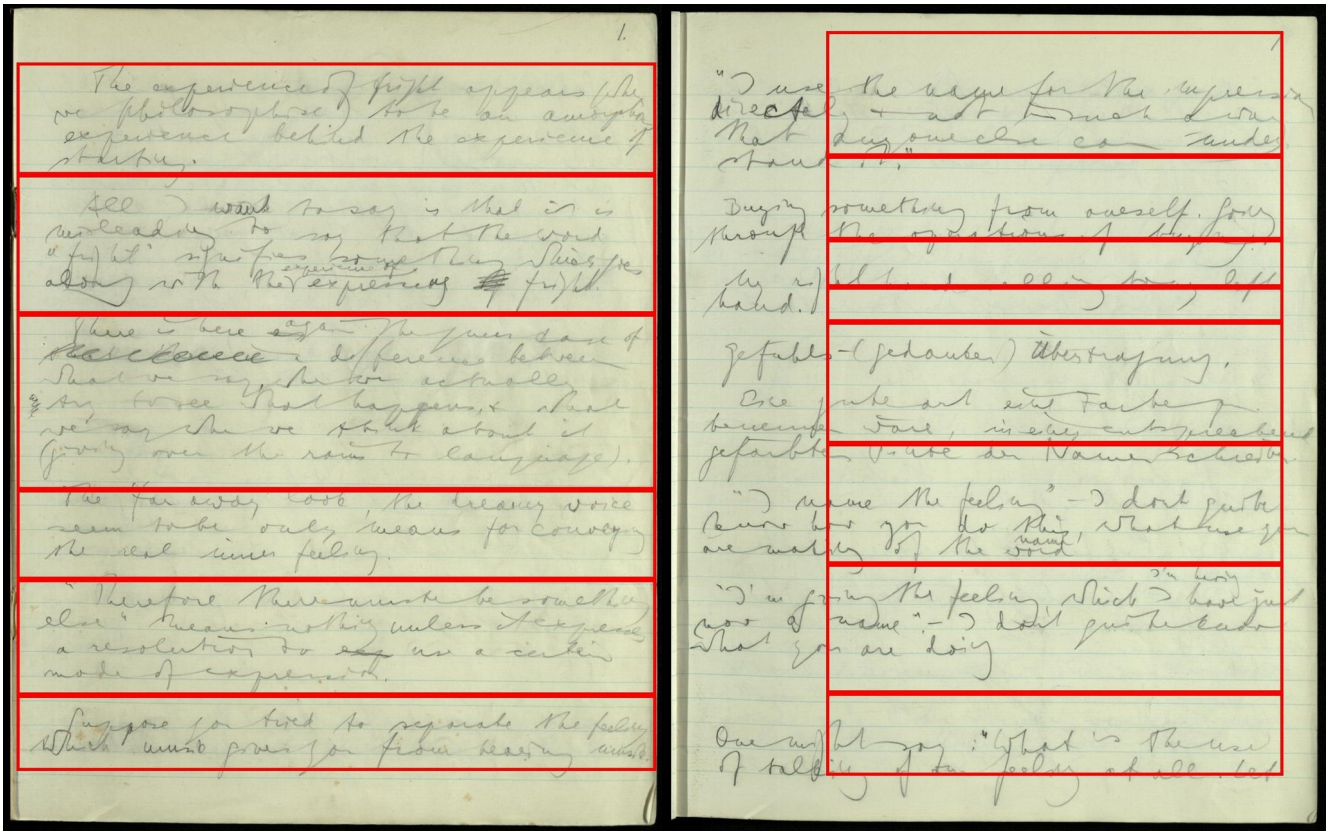


Abb. 16: Eine gute (Ms-148,1r) und eine schlechte Schätzung auf Zeichenbasis (Ms-148,5r)

Über den Befehl „Koordinatenbestimmung“ können nach dem dritten Verfahren Vorschläge für Manuskripte als siebtem Arbeitsschritt erzeugt werden.

Ohne menschliche Nachkorrektur sind mit allen drei Ansätzen keine präzisen Koordinatenvorschläge generierbar. Deshalb wurde über den quelloffenen Anwendungsbereich hinausgehend ein Teil der urheberrechtlich nicht beschränkten Dokumente mit Transkribus bearbeitet.

4.4.4.3. Verwendung von Text2Image unter Transkribus

Das Text2Image-Werkzeug ermöglicht wie oben dargestellt eine automatische Alignierung handgeschriebener Texte mit einer vorhandenen Transkription. Als Eingaben werden die Faksimile sowie dazu korrespondierende, seitenweise transkribierte Textdateien erwartet. Die Textdateien müssen abgesehen von der Dateiendung gleich benannt werden und sollten korrekte Zeilenumbrüche enthalten.[119]

Zunächst kann T2I mit einem existierenden Modell versuchen, erste Zeilen zu alignieren. Gelingt das nicht, so ist die manuelle Verknüpfung unumgänglich. Wurde eine genügend große Menge Zeilen korrekt aligniert, kann die Transkribus-HTR zum Trainieren genutzt werden. Nach einem ersten Durchlauf werden Alignierung und Training wiederholt. Mit dem nun angepassten Modell kann in der Regel ein deutlich größerer Anteil an Zeilen korrekt miteinander verbunden werden. Leere, nicht verknüpfte Zeilen werden automatisch beim Trainingsprozess ignoriert. Dieser sich selbst verbessernde Kreislauf kann vier bis fünfmal gewinnbringend wiederholt werden. Bei einem Versuchsdurchlauf mit Ms-114 wurde nach Erstellung der erforderlichen TXT-Dateien aus der Edition eine Erkennungsrate von circa 75 bis 80 Prozent korrekt alignierter Zeilen erreicht.[120]

Vor Trainingsbeginn muss darauf geachtet werden, dass keine Ausgangsdaten in einer von Wittgenstein benutzten Geheimschrift vorliegen.[121] Dies würde das generierte Modell stören oder sogar gänzlich unbrauchbar machen.

Die fertigen Ergebnisse lassen sich als ALTO-XML Dokumente exportieren. Mit einer Leseroutine können sie in die vorhandene Python-Seitenstruktur eingepasst werden. Die Daten sind ohne weitere Veränderungen mit den Methoden zur Erzeugung der Koordinateninformationen für Typoskripte verwendbar. Auf dieser Grundlage erzeugte Vorschläge sind deutlich präziser als die nur abgeschätzten Koordinaten. (Vgl. Abb. 16)

Text2Image ermöglicht damit eine deutliche Verbesserung der bisher betrachteten Vorgehensweisen. Probleme bei der Bearbeitung mit T2I ergaben sich durch Versionssprünge sowohl server- als auch clientseitig. Insbesondere die im Juni 2019 erfolgte Umstellung von der bisherigen HTR-Version auf ein verbessertes HTR+-System machte das Werkzeug zeitweise unverwendbar. Statt einer Alignierung kamen lediglich Java-Fehlermeldungen zurück. Es blieb nichts anderes als die Fehler den Entwicklern in Innsbruck zu melden. Als weiteres Hindernis muss die sehr rudimentär

119 [O. A.], How to use existing transcriptions to train a Handwritten Text Recognition (HTR) model, Innsbruck, Rostock, 2018, Netz: https://transkribus.eu/wiki/images/6/6f/HowToUseExistingTranscriptions.pdf, Stand: 10.08.2019.

120 In Zusammenarbeit mit dem CITlab durchgeführter Versuch mit Ms-114.

121 Ilse Somavilla, Verschlüsselung in Wittgensteins Nachlass, Innsbruck, 2013, Netz: wittgensteinrepository.org/agora-ontos/article/download/2182/2400, Stand: 10.08.2019, vgl. Geheimschrift-Übersetzer, Netzauftritt: http://wittfind.cis.lmu.de/code/index.html, Stand: 10.08.2019.

ausgestaltete GUI Schnittstelle angemahnt werden. So ist beispielsweise keine Festlegung auf die in der digitalen Edition alignierten Zeilenumbrüche erzwingbar. Leicht ausfilterbare Fehlsegmentierungen müssen deshalb in der Ausgabe belassen werden. Eine Anpassung an eigene Bedürfnisse ist durch die Geheimhaltung des Quelltextes ausgeschlossen.

Einen zusätzlichen Mehrwert der digitalen Editionsarbeit ergibt die Einbindung der übertragenen Daten zur Erstellung eines universellen Handschriftmodells.

Vorerst konnten mit allen beschriebenen Ansätzen weder für Typo- noch für Manuskripte Vorschläge generiert werden, die ohne menschliche Nachkorrektur auskommen. Für diese Aufgabe war ein netzgestützter Nachkorrektor von Nöten.

Zur Einspeisung der erzeugten Vorschläge werden mit dem Befehl Koordinatenbestimmung „coordinates_relative“-Dateien mit dem Dokumentnamen als Prefix im Arbeitsverzeichnis abgelegt. Sie speichern auf Basis der getrennten Siglen als Schlüssel die Koordinateninformation in relativen Positionsangaben unabhängig von der festen Pixelgröße der Ausgangsbilder im JSON-Format. Die gewonnenen Informationen sind auf alle Bilder, die das Verhältnis von Höhe und Breite beibehalten, übertragbar. Als Felder wurden der Koordinatenpunkt des Textbereiches oben links sowie die Höhe und Breite abgespeichert.

Die ebenso vom Wittgensteinarchiv bereitgestellten Norm-HTML-Daten sind auf eine Darstellung im Browser optimiert und können anstelle des geparsten Textes dargestellt werden. Ein Parser extrahiert die notwendigen HTML-Elemente. Zur Einbindung ist der letzte Befehl in der Arbeitskette mit dem Namen „HTML-Integration“ aufzurufen. Die gelesenen Informationen werden in die „coordinates_relative“-Dateien geschrieben. Die fertigen Vorschläge sind nun in dem eigenen Git-Repositorium „coordinates-data“ einspielbar. Nach automatischer Validierung werden sie in die WiTTFind-Datenbank aufgenommen.

4.5. Nachbearbeitung mit dem Korrektor

Um die Ergebnisse der OCR-Arbeit effizient korrigieren zu können war ein geeignetes Nachbearbeitungswerkzeug mit erweitertem Funktionsumfang erforderlich. Ein netzbasiertes System wurde vom ehrenamtlichen CIS-Mitarbeiter Matthias Lindinger in JavaScript implementiert.[122] Insbesondere die Möglichkeit, die Nachkorrektur der circa 20.000 Faksimilebilder auf eine große Anzahl an Nutzer zu verteilen und Bearbeitungskonflikte zu vermeiden, stand bei der Erstellung im Vordergrund.

122 Coordinates-corrector, GitLab-Repositorium, Netz: https://gitlab.cis.uni-muenchen.de/CAST/coordinates-corrector, Stand: 10.08.2019, Koordinaten-Korrektur, Netzauftritt: www.corrector.cis.uni-muenchen.de, Stand: 10.08.2019.

Dafür werden aus einem Dokument Pakete zu zehn Seiten gebildet. Sie sind nach dem Import aus der Datenbank zunächst gesperrt und können vor der Freischaltung zur Korrektur ein letztes Mal auf gravierende Fehler überprüft werden. Außerdem ist es auf diesem Weg möglich, die Menge an freigegebenen Paketen zu kontrollieren. Für zwei verschiedene Korrekturstufen ist ein entsprechender Zustand vorgesehen. Die Korrigierer melden sich mit ihrem Benutzernamen und Passwort an und können ein freies Paket auswählen, das für die anderen Nutzer gesperrt wird.

Eine Korrekturansicht zeigt in der linken Bildschirmhälfte das Faksimile und stellt in Form von gelb hervorgehobenen Koordinatenrechtecken die berechneten Vorschläge dar. Auf der rechten Bildschirmhälfte findet sich die Repräsentation der HTML-Elemente.

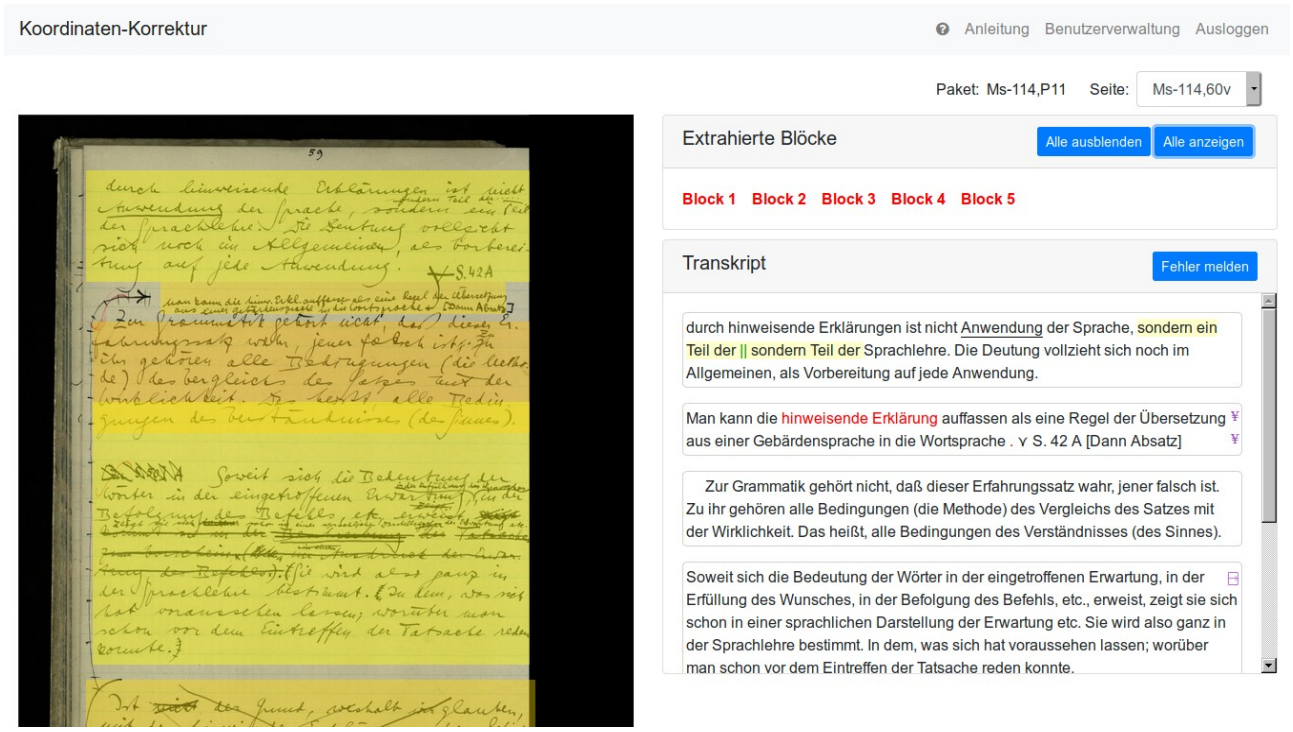


Abb. 17: Ms-114,60v mit von Transkribus-Daten erzeugten Vorschlägen im Korrektor

Sie bilden Unterstreichungen und Hervorhebungen korrekt ab und erlauben eine bessere Orientierung. Gerade bei schwer lesbaren Dokumenten wird die Arbeit deutlich erleichtert. Anderseits können auch Fehler in der digitalen Edition sicher erkannt und über eine entsprechende Schaltfläche gemeldet werden.

Die Fehlermeldungen werden mit Status angelegt und beim Initialisieren auf „offen“ gesetzt. Zweitkorrektoren bekommen Meldungen angezeigt und können deren Zustand verändern. Lassen sich Fehler direkt beheben, zum Beispiel im Falle fehlerhafter OCR-Informationen, ist es dem Zweitkorrektor möglich, Meldungen zu bearbeiten und zu schließen. Andernfalls besteht die Option, die Fehler entweder an die Verantwortlichen des CIS oder der Universität Bergen

weiterzuleiten. Letztlich profitiert damit das Editionsvorhaben selbst von der Erzeugung der geleisteten OCR-Arbeit.

Neben den als einfachen Korrigieren und „Reviewern“ eingestuften Benutzerkonten gibt es als dritte Nutzkategorie Administratoren oder „Admins“. Sie haben zusätzlich Zugriff auf eine Benutzerverwaltung, die neben dem Anlegen neuer Benutzer auch das Sperren von Konten erlaubt. Der Import neuer Dokumente ist ebenso Administratoren vorbehalten.

Der Korrektor wurde vollständig neu erstellt und seither in regem Gedankenaustausch fortlaufend verbessert. Das schon beschriebene Meldungssystem, eine Willkommensseite oder eine Statistik-funktion kamen nachträglich hinzu. Letztere gibt einen Überblick über die Korrekturaktivitäten in der Vergangenheit.

Alle Daten werden in einer MONGO-NoSQL-Datenbank vorgehalten. Sowohl der Dienst für die grafische Benutzeroberfläche als auch die Datenbank laufen auf einem Vermittlungsrechner des CIS. Nach Abschluss der Zweitkorrektur erhalten die Pakete den Zustand fertig. Sind alle Pakete eines Dokuments korrigiert, können die Koordinaten direkt in WiTTFind eingespielt werden. Die Nutzer finden für einen weiteren Teil des Nachlasses Bildausschnitte.

5. Abschluss: OCR – Eine wichtige Hilfe bei der Ermittlung von Koordinatenvorschlägen

OCR-Technologie hat sich für die Verknüpfung von digitaler Edition und Faksimile als ein hilfreiches Instrument gezeigt, um teilautomatisch Koordinatenvorschläge für Bildbereiche auf digitalen Faksimile zu generieren.

Dafür war die Schaffung eigener Werkzeuge im Bereich der Vor- und Nachbearbeitung sowie die Auswahl und Parametrisierung verfügbarer Erkennungssysteme erforderlich. ScanTailor ließ sich aufgrund der Bildgrößenveränderung und der Verschiebung des Textbereichs nicht verwenden. Mit Hilfe von ImageMagick konnte ein dreistufiger Prozess bestehend aus Verkleinerung, Randentfernung und Binarisierung geschaffen werden, der die Bilder für den eigentlichen Erkennungsvorgang aufbereitet. Wegen seiner Robustheit kam Tesseract in der Version 3.04.01 als Erkennungssystem zur Verwendung. Zusammen mit den Editionsdaten wurden mit Hilfe der OCR-Ergebnisse Verfahren erarbeitet, um siglenweise Koordinatenvorschläge mit einem Ähnlichkeits-maß zu berechnen bzw. mit Hilfe des ermittelten Textbereiches auf der Seite anteilig abzuschätzen. Dafür waren beide Datenbestände in eine Datenstruktur einzupassen. Nach Abgleich beider Information wurden die fertigen Vorschläge im Korrektor berichtigt und in WiTTFind integriert.

Im Falle sauberer Typoskripte und akkurater Editionsdaten waren die erzielten Ergebnisse recht vielversprechend. Bei einer weiteren Verbesserung der erarbeiteten Werkzeuge, der Ausgangsdaten wie auch der OCR-Technologie könnte der menschliche Korrekturaufwand weiter reduziert, auf lange Sicht auf den Endanwender übertragen oder sogar weitgehend überflüssig werden. Aus Manuskripten sind mit quelloffenen Mitteln bisher nur mehr oder minder präzise Schätzungen generierbar.

Im derzeitigen Zustand war für beide Bereiche kein Auskommen ohne mehrstufige Nachkorrektur möglich. Dafür wurde der Korrektor von Matthias Lindinger implementiert. Er ermöglicht es, die große Menge an Daten auf viele Schultern zu verteilen. Nach manueller Berichtigung können die Koordinatenvorschläge direkt in den Faksimilereader integriert werden.

Bei Abschluss der vorliegenden Arbeit waren vom Gesamtbestand der ca. 20.000 Seiten 13.000 bearbeitet. Der Arbeitsablauf ist mittlerweile eingespielt und funktioniert zuverlässig. Die automatisierte Installation macht eine Übertragung auf andere Rechner einfach.

Verbesserungspotential gibt es bei der Berechnung von Vorschlägen bei Typoskripten. Hier könnten mit einem Abfangsystem für falsch oder nicht kodierte Zeilenumbrüche in den Editionsdaten und einer Verfeinerung des Alignierungsalgorithmus Verbesserungen erreicht werden.

Die geschaffenen Werkzeuge sind so strukturiert, dass Erkennungsergebnisse verschiedener OCR-Anwendungen in das bestehende System eingespielt werden können. Allerdings sind sowohl die erzeugte Datenstruktur als auch die Vorverarbeitungskette weitgehend auf die speziellen

Bedürfnisse der Wittgensteindaten zugeschnitten. Eine Übertragung auf andere Bestände ist nur nach vorheriger Anpassung möglich. Hier kommt die grundsätzliche Problematik der großen Uneinheitlichkeit von digitalen Editionsprojekten zum Tragen.

Positive Rückkopplungseffekte haben sich in dreierlei Hinsicht ergeben. Erstens profitiert das digitale Editionsprojekt von den Fehlermeldungen der nochmaligen Korrektur. Zweitens können die gewonnen Koordinateninformation in die Bergener Edition aufgenommen werden. Drittens sind aus beiden Beständen Informationen extrahierbar, die sich für die Zwecke der Handschrifterkennung nutzen lassen.

Sollte sich im quelloffenen Bereich ein Handschrifterkennungssystem herausbilden, das mit TXT- und Bilddaten trainiert werden kann, so stünde die Wittgenstein-Edition als mögliches Trainingskorpus zur Verfügung.

Die bereits recht gut funktionierende Alignierung mit dem Text2Image Werkzeug zeigt die großen Möglichkeiten, die sich künftig für eine weitere Automatisierung des Digitalisierungsprozesses bieten. So wurde dargelegt, dass aus den bestehenden Editionsdaten in Kombination mit den zugehörigen Faksimile ein Handschriftmodell ausgebildet werden kann. Denkbar erscheint künftig beispielsweise eine Hervorhebung einzelner Sätze, Wörter oder Zeilen. Demgegenüber stehen bei einer Verwendung von Transkribus die rechtlichen Schwierigkeiten, die sich durch den Zwang der Datenübertragung auf Fremdrechner in Kombination mit nicht vorhandenen Bildrechten ergeben. Hinzu kommt das Fehlen einer Adaptierbarkeit durch die Geheimhaltung des Quelltextes. Eine erneute Erarbeitung von Werkzeugen im quelloffenen Bereich würde bedeuten, eine große Menge Forschungsarbeit erneut leisten zu müssen.

Künstliche Hürden könnte die Schaffung intuitiver Konversionswerkzeuge für verschiedene Ausgabeformate beseitigen.

Hilfreich wäre eine erweiterte Kommandozeilenschnittstelle für die Stapelverarbeitung von ScanTailor. Sie könnte die Arbeit bei künftigen Unternehmungen vereinfachen und vereinheitlichen. Eine Neuimplementierung in Python würde zumindest im Bereich der Vorbearbeitung eine große Erleichterung bedeuten.

Es wurde dargelegt, dass OCR-Technologie ein vielseitiges und dynamisches Arbeits- und Forschungsfeld ist, in dem gerade im Bereich der Handschrifterkennung noch große Verbesserungen zu erwarten sind.

Besonders interessant erscheint die Möglichkeit, bereits vorhandene Transkriptionen zur Erzeugung von Schriftmodellen zu nutzen. Ein universelles Lesewerkzeug in Perfektion ist eine kaum erreichbare Vision. Im Zuge einer noch anstehenden Archivdigitalisierung versprechen weitere

Forschungen jedoch große Erleichterungen bei der für die Wissenschaft und das kulturelle Erbe so zentralen Aufgabe der Retrodigitalisierung.

6. Literatur- und Quellenverzeichnis

6.1. Literaturverzeichnis

- Baierer, Konstantin, Zumstein, Philipp, Verbesserung der OCR in digitalen Sammlungen von Bibliotheken, in: Zeitschrift für Bibliothekskultur, Mannheim, Nr. 2, 2016, S. 72–83, Netz: https://madoc.bib.uni-mannheim.de/41442/1/Baierer-Zumstein-2016.pdf, Stand: 10.08.2019.

- Breuel, Thomas M., The hOCR Microformat for OCR Workflow and Results, in: Proc. ICDAR2007, [o.O.], 2007, S. 1063-1067, Netz: https://www.dfki.de/web/forschung/publikationen/renameFileForDownload?filename=The%20hOCR%20Microformat.pdf&file_id=uploads_381, Stand: 10.08.2019.

- Breuel, Thomas M., The OCRopus open source OCR system, in: Proceedings of SPIE, San Jose, CA 2008, S. 68150F-68150F–15, Netz: http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=812144, Stand: 30.04.2019.

- Ceynowa, Klaus, Von der Aura des Originals zur Immersivität des Digitalen Experimente der Bayerischen Staatsbibliothek im virtuellen Kulturraum, in: Bibliotheken. Innovation aus Tradition, Ceynowa, Klaus (Hg.), Hermann, Martin (Hg.), Berlin, München, 2014, S. 249-257, Netz: https://www.degruyter.com/view/books/9783110310511/9783110310511.249/9783110310511.249.xml, Stand: 31.07.2019.

- Day, Michael, IMPACT best practice guide: Metadata for text digitisation and OCR, [o. O.], 2010, Netz: https://www.digitisation.eu/download/website-files/BPG/OpticalCharacterRecognition-IBPG_01.pdf , Stand: 31.07.2019.

- Deutsche Forschungsgemeinschaft (Hg.), DFG-Praxisregeln „Digitalisierung“, Bonn, 2016, Netz: http://www.dfg.de/formulare/12_151/12_151_de.pdf, Stand: 10.08.2019.

- Fink, Florian, Postcorrection Tool (PoCoTo) Manual, München, 2017, Netz: https://github.com/cisocrgroup/Resources/blob/master/manuals/pocoto-manual.pdf, Stand: 10.08.2019.

- Fink, Florian, Springmann, Uwe, CIS OCR Workshop v1.0. OCR and postcorrection of early printings for digital humanities, München, 2016, Netz: https://github.com/cisocrgroup/OCR-Workshop, https://zenodo.org/record/46571, Stand: 10.08.2019.

- Heyman, Stephen, Google Books: A Complex and Controversial Experiment, in: The New York Times, New York, 28.10.2015, Netzausgabe: https://www.nytimes.com/2015/10/29/arts/international/google-books-a-complex-and-controversial-experiment.html, Stand: 10.08.2019.

- Hodel Tobias, Leifert Gundram, Text2Image matching. How to use existing images and transcripts to produce Automated Text Recognition, Rostock, 2017.

- [O. A.], How to use existing transcriptions to train a Handwritten Text Recognition (HTR) model, Innsbruck, Rostock, 2018, Netz: https://transkribus.eu/wiki/images/6/6f/HowToUseExistingTranscriptions.pdf, Stand: 10.08.2019.

- [O. A.], How to train a HTR model in Transkribus, Innsbruck, 2018, Netz: https://transkribus.eu/wiki/images/3/34/HowToTranscribe_Train_A_Model.pdf, Stand: 10.08.2019.

- [O. A.], IMPACT, [o. J.], Netz: https://www.frakturschrift.com/en:projects:impact, Stand: 10.08.2019.

- Jackson, Joab, Google: 129 Million Different Books Have Been Published, in: PCWorld, [o. O.], 06.08.2010, Netzausgabe: https://www.pcworld.com/article/202803/google_129_million_different_books_have_been_published.html, Stand: 10.08.2019.

- Kallio, Maria, Seaward, Louise, Transkribus: Handwritten Text Recognition technology for historical documents, Netz: https://dh2017.adho.org/abstracts/649/649.pdf, Stand: 10.08.2019.

- [O. A.], Ludwig Wittgenstein, in: Wien Geschichte Wiki, Netz: https://www.geschichtewiki.wien.gv.at/Wien_Geschichte_Wiki?curid=11436, Stand: 31.7.2019.

- Mühlberger, Günter, Die automatisierte Volltexterkennung historischer Handschriften als gemeinsame Aufgabe von Archiven, Geistes- und Computerwissenschaftlern. Das Modell einer zentralen Transkriptionsplattform als virtuelle Forschungsumgebung, in: Becker, Irmgard Christa, (Hg.), Oertel, Stephanie (Hg.), Digitalisierung im Archiv. Neue Wege der Bereitstellung des Archivguts. Beiträge zum 18. Archivwissenschaftlichen Kolloquium der Archivschule Marburg, Marburg an der Lahn, 2015, S. 87-115, Netz: https://www.academia.edu/7451967/Die_automatisierte_Volltexterkennung_historischer_Handschriften_als_gemeinsame_Aufgabe_von_Archiven_Geistes-_und_Computerwissenschaftlern._Das_Modell_einer_zentralen_Transkriptionsplattform_als_virtuelle_Forschungsumgebung, Stand: 10.08.2019.

- Mühlberger, Günter, H2020 Project READ. Recognition and Enrichment of Archival Documents. 2016-2019, Innsbruck, 2019, Netz: https://www.academia.edu/22653102/H2020_Project_READ_Recognition_and_Enrichment_of_Archival_Documents_-_2016-2019, Stand: 10.08.2019.

- Nousiainen, Sami, Report on File Formats for Hand-written Text Recognition (HTR) Material, Network, Helsinki, 2016, Netz: https://documents.icar-us.eu/documents/2016/12/report-on-file-formats-for-hand-written-text-recognition-htr-material.pdf, Stand: 10.08.2019.

- [O. A.], OCR in: Fischer, Peter (Hg.), Hofer, Peter (Hg.), Lexikon der Informatik, Heidelberg, Luzern, 201015, https://epdf.pub/lexikon-der-informatik-15-auflage.html, Stand: 31.07.2019, S. 626.

- Ohlsen, Victor, Optical Character and Symbol Recognition using Tesseract, Luleå, 2016, Netz: http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1019846&dswid=2838, Stand: 10.08.2019.

- [O.A.], Optimal image resolution (dpi/ppi) for Tesseract 4.0.0 and eng.traineddata?, Netz: https://groups.google.com/forum/#!msg/tesseract-ocr/Wdh_JJwnw94/24JHDYQbBQAJ, Stand: 10.08.2019.

- Pletschacher, S., Antonacopoulos, A., The PAGE (Page Analysis and Ground-Truth Elements) Format Framework, in: Proceedings of the 20th International Conference on Pattern Recognition (ICPR2010), Istanbul, Turkey, August 23-26, 2010, in: IEEE-CS Press, S. 257-260.

- Puppe, Frank, Reul, Christian, Springmann Uwe, Wick, Christoph, State of the Art Optical Character Recognition of 19th Century Fraktur Scripts using Open Source Engines, Würzburg, 2018, Netz: https://arxiv.org/abs/1810.03436, Stand: 10.08.2019.

- Puppe, Frank, Reul, Christian, Wick, Christoph, Calamari. A High-Performance Tensorflow-based Deep Learning Package for Optical Character Recognition, Würzburg, 2018, Netz: www.arxiv.org/pdf/1807.02004, Stand: 10.08.2019.

- Reul, Christian, OCR4all. An Open Source Tool Providing a Full OCR Workflow, Würzburg, 2019.

- Röhrer Ines, Ullrich, Sabine, Weltkulturerbe international digital. Erweiterung der Wittgenstein Advanced Search Tools durch Semantisierung und neuronale maschinelle Übersetzung, München, 2019, Netz: https://www.cis.uni-muenchen.de/kurse/max/homepage/poster/posterdhd_2019.pdf, Stand: 10.08.2019.

- Smith, Ray, An overview of the Tesseract OCR Engine in: Proc. 9th IEEE Intl. Conf. on Document Analysis and Recognition, [o. O.], 2007, S. 629–633, Netz: https://static.googleusercontent.com/media/research.google.com/de//pubs/archive/33418.pdf, Stand: 10.08.2019.

- Somavilla, Ilse, Verschlüsselung in Wittgensteins Nachlass, Innsbruck, 2013, Netz: wittgensteinrepository.org/agora-ontos/article/download/2182/2400, Stand: 10.08.2019.

- Springmann, Uwe, OCR für alte Drucke, in: Gesellschaft für Informatik e.V. (Hg.), Informatik-Spektrum Bd. 39, Heidelberg, 2016 S. 459-462, Netz: http://springmann.net/papers/2016-10-17-OCR-alte-Drucke.pdf, Stand: 10.08.2019.

- Taycher, Leonid, Books of the world, Stand: up and be counted! All 129,864,880 for you, in: Google Books Search, [o. O.], 05.08.2010, Netz: https://booksearch.blogspot.com/2010/08/books-of-world-stand-up-and-be-counted.html, Stand: 10.08.2019.

- [O. A.], The Wittgenstein Archives at the University of Bergen (WAB), Bergen in Norwegen, Netz: http://wab.uib.no/, Stand: 10.08.2019.

- Weil, Stefan, Zumstein, Philipp, Mit freier Software Text in Digitalisaten erkennen. OCR-Praxis an der UB Mannheim, Mannheim, 2016, Netz: https://www.kitodo.org/fileadmin/groups/kitodo/Dokumente/UBMannheim_OCR-Freie-Software.pdf, Stand: 10.08.2019.

- [O. A.], Wittgenstein's Nachlass. The Bergen Electronic Edition (BEE), Bergen in Norwegen, Netz: http://wab.uib.no/wab_BEE.page, Stand: 10.08.2019.

- [O. A.], 2016 PAGE XML Format for Page Content, Manchester, 2016, Netz: https://www.primaresearch.org/schema/PAGE/gts/pagecontent/2016-07-15/Simple%20PAGE%20XML%20Example.pdf, Stand: 10.08.2019.

6.2. Netzauftritte und Quelltextrepositorien

- Abbyy Finereader, Netzauftritt: https://www.abbyy.com/de-de/finereader/, Stand: 10.08.2019.
- ALTO XML, GitHub-Repositorium, Netz: https://github.com/altoxml, Stand: 10.08.2019, Sami Nousiainen, Report on File Formats, Helsinki, 2016.
- Awesome OCR, GitHub-Repositorium, Netz: https://github.com/kba/awesome-ocr#ocr-file-formats, Stand: 10.08.2019.
- Calamari-OCR, GitHub-Repositorium, Netz: https://github.com/Calamari-OCR, Stand: 10.08.2019.
- Coordinates-corrector, GitLab-Repositorium, Netz: https://gitlab.cis.uni-muenchen.de/CAST/coordinates-corrector, Stand: 10.08.2019.
- Coordinates-data, GitLab-Repositorium, Netz: https://gitlab.cis.uni-muenchen.de/CAST/coordinates-data, Stand: 10.08.2019.
- Coordinates-extractor, GitLab-Repositorium, Netz: https://gitlab.cis.uni-muenchen.de/CAST/coordinates-extractor, Stand: 22.08.2019.
- Digital Humanities im deutschsprachigen Raum e.V., Netzauftritt: https://dig-hum.de/, Stand: 31.07.2019.
- Faksimilereader, Netzauftritt: http://reader.wittfind.cis.lmu.de/, Stand: 10.08.2019.
- Geheimschrift-Übersetzer, Netzauftritt: http://wittfind.cis.lmu.de/code/index.html, Stand: 10.08.2019.
- GImageReader, GitHub-Repositorium, Netz: https://github.com/manisandro/gImageReader, Stand: 10.08.2019.
- GIT-Lab CIS: https://gitlab.cis.uni-muenchen.de, Stand: 10.08.2019.
- GT4HistOCR, Zenodo-Record, Netz: https://zenodo.org/record/1344132, Stand: 10.08.2019.
- HOCR-Spezifikation, Netzauftritt: http://kba.cloud/hocr-spec/1.2/, Stand: 10.08.2019.
- ImageMagick, Netzauftritt: https://imagemagick.org/script/convert.php, Stand: 10.08.2019.
- Koordinaten-Korrektur, Netzauftritt: www.corrector.cis.uni-muenchen.de, Stand: 10.08.2019.
- Kraken, GitHub-Repositorium, Netz: https://github.com/mittagessen/kraken, Stand: 10.08.2019.
- Levenshtein-Algorithmus, Netzauftritt: http://www.levenshtein.de/, Stand: 10.08.2019.
- LXML, Netzauftritt: https://lxml.de/, Stand: 10.08.2019.
- Münchner Digitalisierungszentrum, Netzauftritt: https://www.digitale-sammlungen.de/, Stand: 14.08.2019.
- Ocropus, GittHub-Repositorium, Netz: https://github.com/jkrall/ocropus/tree/master/ocropus, Stand: 10.08.2019.
- Ocropus2, GitHub-Repositorium, Netz: https://github.com/tmbdev/ocropy2, Stand: 10.08.2019.

- Ocropus3, GitHub-Repositorium, Netz: https://github.com/NVlabs/ocropus3, Stand: 10.08.2019.
- Ocropy, GitHub-Repositorium, Netz: https://github.com/tmbdev/ocropy, Stand: 10.08.2019.
- OCR-Conversion-scripts, GitHub-Repositorium, Netz: https://github.com/cneud/ocr-conversion-scripts, Stand: 10.08.2019.
- OCR-D, Koordinierte Förderinitiative zur Weiterentwicklung von Verfahren der Optical Character Recognition (OCR), Netzauftritt: http://ocr-d.de/, Stand: 10.08.2019.
- OCR-gt, GitHub-Repositorium, Netz: https://github.com/cneud/ocr-gt, Stand: 10.08.2019.
- OCR4all, GitHub-Repositorium, Netz: https://github.com/OCR4all, Stand: 10.08.2019.
- Pattern Recognition & Image Analysis Reasearch Lab, Universität Salford, Manchester, Netzauftritt: https://www.primaresearch.org/tools/TesseractOCRToPAGE, Stand: 10.08.2019.
- Pattern Recognition & Image Analysis Reasearch Lab, Universität Salford, Manchester, Netzauftritt: https://www.primaresearch.org/tools/PAGEConverterValidator, Stand: 10.08.2019.
- READ. Recognition and Enrichment of Archival Documents, Netzauftritt: https://read.transkribus.eu/, Stand: 10.08.2019.
- ScanTailor, GitHub-Repositorium, https://github.com/scantailor/scantailor/blob/master, Stand: 10.08.2019.
- ScanTailor, Netzauftritt, https://scantailor.org/, Stand: 10.08.2019.
- ScanTailor Advanced, GitHub-Repositorium, https://github.com/4lex4/scantailor-advanced, Stand: 10.08.2019.
- Scantailor_enhanced, GitHub-Repositorium, Netz: https://gist.github.com/tristan-k/2700275, Stand: 10.08.2019.
- Tesseract-OCR, GitHub-Reporsitorium, Netz: https://github.com/tesseract-ocr/tesseract/blob/master, Stand: 10.08.2019.
- Text Encoding Initiative (TEI), Netzauftritt: https://tei-c.org/, Stand: 10.08.2019.
- The Wittgenstein Archives, Bergen, Netz: http://wab.uib.no/, Stand: 10.08.2019.
- WiTTFind, Netzauftritt: http://wittfind.cis.uni-muenchen.de/, Stand: 10.08.2019.
- Wittgensteinsource, Netzauftritt: http://www.wittgensteinsource.org/, Stand: 10.08.2019.
- Wittgenstein-faksimilie-download, GitLab-Repositorium, Netz: https://gitlab.cis.uni-muenchen.de/wast/wittgenstein-faksimile-download/blob/Bildoptimierung-issue-945, Stand: 10.08.2019.
- Witt-data, GitLab-Repositorium, Netz: https://gitlab.cis.uni-muenchen.de/wast/witt-data, Stand: 10.08.2019.

7. Abbildungsverzeichnis

8. Abkürzungsverzeichnis

BEE	=	Bergen Electronic Edition
CIS	=	Centrum für Informations- und Sprachverarbeitung, LMU
CITlab	=	Computational Intelligence Technology Lab, Universität Rostock
DHD	=	Digital Humanities im deutschsprachigen Raum e.V.
DFG	=	Deutsche Forschungsgemeinschaft
HP	=	Hewlett-Packard
HTR	=	Handwritten Text Recognition
LMU	=	Ludwig-Maximilians-Universität München
Ms	=	Manuskript
OCR	=	Optical Character Recognition
OZE	=	Optische Zeichenerkennung
TEI	=	Text Encoding Inititative
Ts	=	Typoskript
T2I	=	Text2Image-Werkzeug der Universität Rostock, vgl.: Anmerkung
WAB	=	Wittgenstein Archives Bergen

9. Anhang

Inhalt der beiliegenden CD-Rom:

- Identische elektronische Fassung der Arbeit im ODT- und PDF-Format
- ZIP-Archiv mit dem erstellten Quelltext

Erklärung zur Erstellung der Bachelorarbeit

Hiermit versichere ich,

- dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen sowie Hilfsmittel benutzt habe.
- dass die elektronische Fassung und die Druckfassung der Arbeit identisch sind.

....

(Ort, Datum) (Unterschrift)