Über LERA und die Realisierung wesentlicher Anforderungen an digitale Werkzeuge zur Kollationierung verschiedener Textfassungen eines Werkes

Dissertation

zur Erlangung des
Doktorgrades der Naturwissenschaften (Dr. rer. nat.)

der Naturwissenschaftlichen Fakultät III
Agrar- und Ernährungswissenschaften,
Geowissenschaften und Informatik
der Martin-Luther-Universität Halle-Wittenberg

vorgelegt von
Marcus Pöckelmann

Gutachter:innen
Prof. Dr. Paul Molitor
Prof. Dr. Andrea Rapp

Tag der Verteidigung: 25.06.2024

Zusammenfassung

In dieser Dissertation wird das digitale Kollationierungstool LERA – ein Akronym für *Locate, Explore, Retrace and Apprehend complex text variants* – vorgestellt, welches in mehrjähriger Eigenentwicklung im Rahmen interdisziplinärer Forschungsprojekte am Institut für Informatik der Martin-Luther-Universität Halle-Wittenberg entstand. Kollationierung ist der Prozess, mehrere Textfassungen eines Werkes miteinander zu vergleichen, um so die Gemeinsamkeiten und Unterschiede in Form von Textvarianten herauszuarbeiten, die wiederum als Bestandteil wissenschaftlicher Editionen untersucht und publiziert werden. Digitale Editionen gelten dabei als ein „Kronjuwel“ der Digital Humanities, jener Disziplin, welche traditionelle Methoden der Geisteswissenschaften durch die Adaption im Digitalen modernisiert sowie durch neuartige Ansätze erweitert. Als Teil dieser Dissertation werden anhand verschiedener Editionsprojekte sechs allgemeine Anforderungen an Software zur Unterstützung der Kollationierung herausgearbeitet und mit bestehenden Methoden und Werkzeugen dieser Domäne abgeglichen. Die dabei aufgedeckten Diskrepanzen zwischen den Anforderungen und den verfügbaren Werkzeugen motivieren die Entwicklung eines eigenen. Wie im Detail ausgeführt wird, ermöglicht LERA den direkten Vergleich von sehr vielen Textfassungen nach einem gleichrangigen Vergleichsprinzip, die Kollationierung sehr langer Werke und den Umgang mit verschiedenartigen, komplexen Textvarianten. Dabei wurde ein nutzerfreundliches, interaktives und generisches System realisiert. Im Rahmen der Dissertation wird neben einem Überblick über LERAs verschiedene Funktionalitäten mit je einem eigenen Kapitel detailliert auf die zwei Schwerpunkte der Innovation eingegangen: Die algorithmische Berechnung von Textvarianten und die visuelle Exploration dieser. Die Güte der implementierten Lösungen wird durch LERAs mittlerweile breite Anwendung in einer Vielzahl diverser Editionsprojekte durch nationale und internationale Forschende bekräftigt.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die Digitalisierung verändert, erweitert und schafft neue Methoden und Arbeitsweisen in allen Wissenschaftsbereichen. So auch in den Geisteswissenschaften, wo mit den *Digital Humanities* ein thematisch sehr breites, interdisziplinäres Gebiet entstanden ist, von dem mitunter von einer eigenen Disziplin gesprochen wird [88]. Als ein „Kronjuwel" der Digital Humanities gelten dabei digitale Editionen [114], wie bereits hunderte realisierte Projekte [48] sowie eine Vielzahl laufender (siehe Abschnitt 2.1) zeigen.

Die Editionsphilologie selbst ist eine historisch gewachsene und facettenreiche Disziplin, deren Wurzeln bis in die griechische Antike zurück reichen [106, S.21] und die ein breites Spektrum von Zielen verfolgt. Editionen können demnach ganz unterschiedliche Formen aufweisen und Zwecke erfüllen.
Laut einer Definition von Sahle [77, S.239] versteht man unter einer Edition die „erschließende Wiedergabe historischer Dokumente". Mit *Dokumenten* sind meist Texte gemeint und diese Dissertation beschränkt sich darauf, wenn gleich auch physische Objekte oder musikalische Stücke im Fokus einer Edition stehen können. Da es sich in der Regel um *historische* Dokumente handelt, ist eine Distanz zwischen dem heutigen Verständnis und der historischen Wirklichkeit zu überbrücken. Neben der eigentlichen *Wiedergabe*, also der möglichst vollständigen Repräsentation des Gegenstands der Edition, besteht daher ein Bedarf an Einordnung und Erläuterung. Die *Erschließung* durch die Editor:innen umfasst entsprechend eine ganze Reihe kritischer Operationen, die mit der Auswahl und (äußeren) Beschreibung der zu edierenden Objekte beginnt, weiter über ihre Transkription und den damit einhergehenden Entscheidungen, hin zur Identifikation von referenzierten Dingen,

der Anfertigung von Sachanmerkungen und schließlich der eigentlichen Textkritik reicht.
Eine Edition ist auch immer der Versuch, eine substanzielle Grundlage für die weitere Forschung zu schaffen. Oft wird dazu entweder der überlieferte Textbestand dokumentiert oder ein Text produziert, der möglichst der Intention der Autor:innen entspricht, das heißt ohne Verfälschung von Kopierer:innen, Zensor:innen, Verlagen (Setzer:innen und Herausgeber:innen), Erb:innen oder Fehlern der Autor:innen selbst [122, 114]. In beiden Fällen ist ein Abgleich der verschiedenen Fassungen des Textes zum Herausarbeiten von Textvarianten essentiell, die sogenannte Kollationierung.

Die Bestimmung und Analyse von Textvarianten ist das Hauptaugenmerk dieser Arbeit. Eine Textvariante beschreibt einen Unterschied zwischen verschiedenen Fassungen eines Textes. Dass mehrere Fassungen eines Textes existieren, kann viele Gründe haben. Für handschriftliche Werke sind das beispielsweise Abschriften des ursprünglichen Manuskripts beziehungsweise einer früheren Abschrift[1], die Verschriftlichung von Volksmärchen zu unterschiedlichen Zeiten und an verschiedenen Orten[2] oder auch die Übersetzung eines Textes von verschiedenen Personen[3]. Mit dem Buchdruck entstanden weitere forschungsrelevante Formen, wie verschiedene, teils umfangreich überarbeitete Druckauflagen eines Werkes[4].
Oftmals überarbeiten Autor:innen selbst ihren Text, sodass die Beschränkung auf genau eine Textfassung, wie die Erstausgabe oder die Ausgabe letzter Hand, zum Verstehen eines Werks zu kurz greift, wie Price in [122] darlegt: So stecke in späteren Versionen mehr Arbeit und sie seien gewissenhafter durchdacht sowie präziser ausformuliert als beispielsweise die ersten Entwürfe. Andererseits sei nicht klar, ob die Autor:innen am Lebensende wirklich auf ihrem geistigen Leistungshoch sind. Zudem würden mit der Zeit mitunter Bedenken und Zensur in den Text einfließen[5].

[1] Zum Beispiel die *Wundarznei* des Heinrich von Pfalzpaint [84].

[2] Zum Beispiel das Märchen vom *Rotkäppchen* [78].

[3] Zum Beispiel verschiedene Übersetzungen der *Bibel* [91].

[4] Zum Beispiel die *Histoire des deux Indes* von Guillaume Thomas François Raynal [23].

[5] Aus dem Englischen: „the final version of a work is often stronger — more fully developed, more carefully considered, more precisely phrased — than an early or intermediate draft of it. But for poets, novelists, and dramatists whose work may span decades, there is real question about the wisdom of relying on last choices. Are people at their sharpest, most daring, and most experimental at the end of life when energies (and sometimes clarity) fade and other signs of age begin to show?“ [122].

Je nach Ursprung und Intention fallen die Textvarianten unterschiedlich in Art und Umfang aus und reichen beispielsweise von einer veränderten Typografie oder Kommasetzung hin zur kompletten Umstrukturierung oder inhaltlichen Überarbeitung des Textes (siehe Abschnitt 2.2). Welche Textvarianten für Forschende von Interesse sind, hängt wiederum vom konkreten Editionsvorhaben und dessen Forschungsfragen ab. Im einfachsten Fall wird versucht, der Leserschaft den *besten* Text möglichst nahe an den Vorstellungen der ursprünglichen Autor:innen zu präsentieren. Dazu gehören Fehlerbereinigung, also das Entfernen ungewollter Schreib- oder Druckfehler, die Normierung von Orthographie oder Grammatik und mitunter die Anpassung an eine moderne Sprachstufe, beispielsweise im Zuge einer Rechtschreibreform. In der Editionsphilologie werden aber auch komplexere Aufgaben verfolgt, die im Zusammenhang mit dem Herausarbeiten von Textvarianten stehen. Dazu gehören die Rekonstruktion eines nicht überlieferten Urtextes, die Filiation – also das Aufzeigen des Abstammungsverhältnisses der Textfassungen –, die Textgenese – also das Nachzeichnen der Entstehungs- und Änderungshistorie – und schließlich die Einordnung von Textvarianten in den historischen Kontext. Auf die Hintergründe und Forschungsziele bei der Kollationierung wird in Kapitel 2 anhand konkreter Editionsprojekte detaillierter eingegangen.

Es gibt seit vielen Jahren etablierte Ansätze und Werkzeuge zur Unterstützung der Editionsphilologie im Allgemeinen und der Kollationierung im Besonderen, welche allerdings wesentliche Anforderungen für viele Editionsprojekte nicht erfüllen, wie in Kapitel 3 aufgezeigt wird.

Im Rahmen dieser Dissertation wird die eigens entwickelte Arbeitsumgebung LERA – ein Akronym für *Locate, Explore, Retrace and Apprehend complex text variants* – zum Auffinden und Analysieren von Textvarianten vorgestellt. LERA wurde anhand wesentlicher Anforderungen aus konkreten Editionsvorhaben entwickelt und entstand in vieljähriger Forschungstätigkeit in der eHumanities-Arbeitsgruppe von Prof. Dr. Paul Molitor und Dr. Jörg Ritter an der Martin-Luther-Universität Halle-Wittenberg.

Diese Dissertation zeigt und diskutiert die theoretische Modellierung sowie die praktische Umsetzung wesentlicher Funktionalitäten von LERA im Abgleich mit den zuvor aufgeschlüsselten geisteswissenschaftlichen Anforderungen sowie dem Gothenburg-Modell, welches den Prozess der automatischen Kollationierung in verschiedene Schritte einteilt (Kapitel 4). Ein Hauptaugenmerk dieser Arbeit liegt dabei

auf zwei Kernfunktionen von LERA, welche in je einem eigenen Kapitel diskutiert werden. Das ist zum Ersten die algorithmische Berechnung von Textvarianten (Kapitel 5), welche über einen mehrstufigen Ansatz durch Einteilung der Texte in Fassungs-, Segment- und Tokenebene realisiert wird. LERA implementiert zudem das gleichrangige Vergleichsprinzip, nach welchem jede Textfassung mit jeder verglichen wird. Darin verbirgt sich wiederum ein NP-vollständiges Problem, welchem für die praktische Umsetzung mit einer Heuristik begegnet wird. Zum Zweiten wird die explorative Analyse der Textvarianten mittels LERA in Kapitel 6 ausführlich beschrieben. Dazu wurden drei interaktive Komponenten in LERA integriert und miteinander kombiniert: eine Suchfunktion, die Übersichts- und Navigationsvisualisierung CATview und stabile Wortwolken. Zudem werden mit der in LERA integrierten Korpussynopse drei interaktive Visualisierungen vorgestellt, die eine explorative Analyse auf Fassungsebene ermöglichen. Ein zusammenfassendes Kapitel mit Angaben zur Nutzerbasis von LERA sowie einem Ausblick auf mögliche Erweiterungen der Software schließen diese Dissertation ab.

Kapitel 2

Bedeutung des Textvergleichs in den Philologien

Dieses Kapitel vertieft zunächst anhand einiger konkreter Projekte und Editionsvorhaben die bereits zuvor im Einleitungskapitel angeschnittene Motivation, verschiedene Textfassungen eines Werkes miteinander zu vergleichen. Gleichzeitig wird dabei deutlich, dass Software diesen Prozess sehr erleichtern kann. Trotz der Unterschiedlichkeit der Editionsvorhaben zeichnen sich wiederkehrende Problemklassen ab, die in Abschnitt 2.2 heraus gearbeitet und benannt werden. Diese bilden wiederum die Grundlage für die sechs in Abschnitt 2.3 formulierten Anforderungen an Kollationierungssoftware und damit den roten Faden, der sich durch diese Dissertation zieht. Das Kapitel schließt mit einem ergänzenden, vierten Abschnitt, in dem einige der in dieser Dissertation genutzten Begrifflichkeiten definiert sowie ihre formale Notation eingeführt werden.

2.1 Editionsvorhaben am Beispiel

Im Folgenden wird die Bedeutung der Analyse verschiedener Textfassungen anhand einiger Werke und der dazugehörigen Editionsvorhaben verdeutlicht.

2.1.1 Abbe Raynals Histoire des deux Indes

Die *Histoire philosophique et politique des établissements et du commerce des Européens dans les deux Indes*, kurz *Histoire des deux Indes*, ist ein umfassendes und

wichtiges Werk aus der Zeit der französischen Aufklärung. Der Hauptautor Guillaume Thomas François Raynal (* 1713 in Lapanouse; † 1796 in Passy) schreibt darin über die europäische Kolonialpolitik. Das Werk gliedert sich dabei in 19 einzelne Bücher, welche die Kolonialisierung anhand einzelner Regionen beziehungsweise durch einzelne europäische Mächte beschreiben. So umfasst beispielsweise Livre VI die spanische Herrschaft in Mexiko. Die einzelnen Bücher enthalten meist die Geschichte der Kolonie sowie Angaben zur Geografie, Bodenschätzen, Flora und Ausfuhrwaren, aber auch unterhaltsame Anekdoten und eine kritische Aufarbeitung der Ereignisse, siehe [23]. Die *Histoire des deux Indes* besitzt eine komplexe Druckgeschichte in Folge diverser umfassender Überarbeitungen durch den Autor, wobei vier grundsätzlich unterschiedliche Fassungen des Textes existieren: Die anonym veröffentlichten und schnell verbotenen Fassungen von 1770 (Erstausgabe) und 1774, die wohl bekannteste Fassung von 1780 und eine 1820 posthum erschienene vierte Fassung, in welche die Ergänzungen von Raynals Handexemplar der dritten Fassung eingearbeitet wurden.

Die *Histoire des deux Indes* ist Untersuchungsgegenstand zweier Projekte. Eines ist ein klassisches Editionsvorhaben zur Erstellung einer kritischen Edition in Buchform unter der Direktion von Anthony Strugnell, Gianluigi Goggi und Kenta Ohji. Die ersten drei Bände, die jeweils fünf Bücher der *Histoire des deux Indes* umfassen, sind 2010 [123], 2018 [124] und 2020 [125] erschienen. Die kritische Edition reproduziert jeweils die Fassung von 1780 als Basistext mit begleitenden Stellenkommentaren und führt die Unterschiede zu den Fassungen von 1770 und 1774 in einem Variantenapparat auf. Hinzu kommt ein Anhang mit den Unterschieden der vierten Fassung von 1820.

Auch in Abgrenzung zu dieser klassischen Edition wurde Livre IV der *Histoire des deux Indes* als eines von zwei Fallbeispielen im Rahmen des interdisziplinären Forschungsprojekts *Semi-automatische Differenzanalyse von komplexen Textvarianten (SaDA)* [22, 95] untersucht. Ziel von *SaDA* war dabei philologische Teilprozesse der editorischen Arbeit zu identifizieren, die mittels Methoden der Informatik möglichst generisch für verschiedene Editionsvorhaben unterstützt werden können. Das Auffinden, die übersichtliche Visualisierung sowie die explorative Analyse von Textvarianten zwischen den Fassungen standen im besonderen Fokus.

Die *Histoire des deux Indes* bringt besondere Herausforderungen mit sich. Zunächst sind es sehr umfangreiche Druckauflagen, selbst wenn nur ein einzelnes Buch betrachtet wird. So beinhaltet Livre VI, das Fallbeispiel im *SaDA*-Projekt, in der Fas-

sung von 1770 knapp 250 Absätze mit insgesamt rund 28.000 Wörtern und in der Fassung von 1820 gut 420 Absätze mit rund 52.000 Wörtern. Zwar sind der Hauptautor und die Chronologie bekannt, dennoch unterscheiden sich die Fassungen substanziell voneinander. Die komplexen Umarbeitungen in Form von Erweiterungen, Streichungen und kompletten Überarbeitungen ganzer Absätze verleihen jeder Fassung dabei ein „eigenes Recht“ [22, S.3]. Dies wird im Folgenden an vier Beispielen veranschaulicht, von denen drei aus Bremer et al. [23] übernommen wurden, wo sie auch näher diskutiert werden. Der Umfang der Überarbeitung zeigt sich beispielsweise direkt am Anfang von Livre VI, wo der ursprüngliche einleitende Absatz von 1770 zur aktuellen politischen Situation Spaniens in der Fassung von 1774 auf neun Druckseiten hin zu einem detaillierten Vergleich antiker und moderner Geschichte erweitert wird, bevor beide Fassungen schließlich auf Christoph Kolumbus zu sprechen kommen. Auch innerhalb eines Absatzes treten häufig inhaltliche Erweiterungen auf, wie der in Abbildung 2.1 veranschaulichte Textvergleich der vier Fassungen zur Begegnung von Montezuma II. und Hernán Cortés verdeutlicht.

Aus editionsphilologischer Sicht besteht ein Interesse darin, die komplexe Textgenese nachzuvollziehen und abzubilden, sodass schließlich geschichts- und kulturwissenschaftliche Fragestellungen mit der Edition beantwortet werden können. So sind vor allem die inhaltlichen Änderungen, beispielsweise zu den Wissensbeständen und Einordnungen des Autors, im Zuge der Überarbeitung interessant. Diese Erweiterungen enthalten oft zusätzliche Informationen; so die erweiterte Angabe zu Kolumbus Expedition von 1492: „Mit diesem kleinen Geschwader, dessen Bewaffnung keine hunderttausend Franken kostete, [...]“ (aus dem Französischen: „Sur cette foible eſcadre, dont l’armement ne coûtoit pas cent mille francs, [...]“), welche in der Fassung von 1780 ergänzt wurde. Raynal bringt mitunter aber auch mehr Emotionen im Werk unter, wie ein drittes in [23] diskutiertes Beispiel zeigt: Ein bereits in drastischen Worten formulierter Abschnitt zur Ausbeutung der Eingeborenen und ihrer daraus resultierenden Lebenswirklichkeit, erweitert der Autor in der Fassung von 1780 um den Zusatz: „Ich muss hier einen Moment innehalten. Meine Augen sind voller Tränen, und ich kann nicht mehr sehen, was ich schreibe.“ (aus dem Französischen: „Il faut que je m’arrête ici un moment. Mes yeux ſe rempliſſent de larmes, & je ne vois plus ce que j’écris.“).

Abseits der inhaltlichen Unterschiede kommen teils geänderte Normierungen der französischen Sprache sowie verschiedene Schreibweisen von Eigennamen als Textvarianten hinzu. Unter anderem sind das wiederkehrende – also systematische – Un-

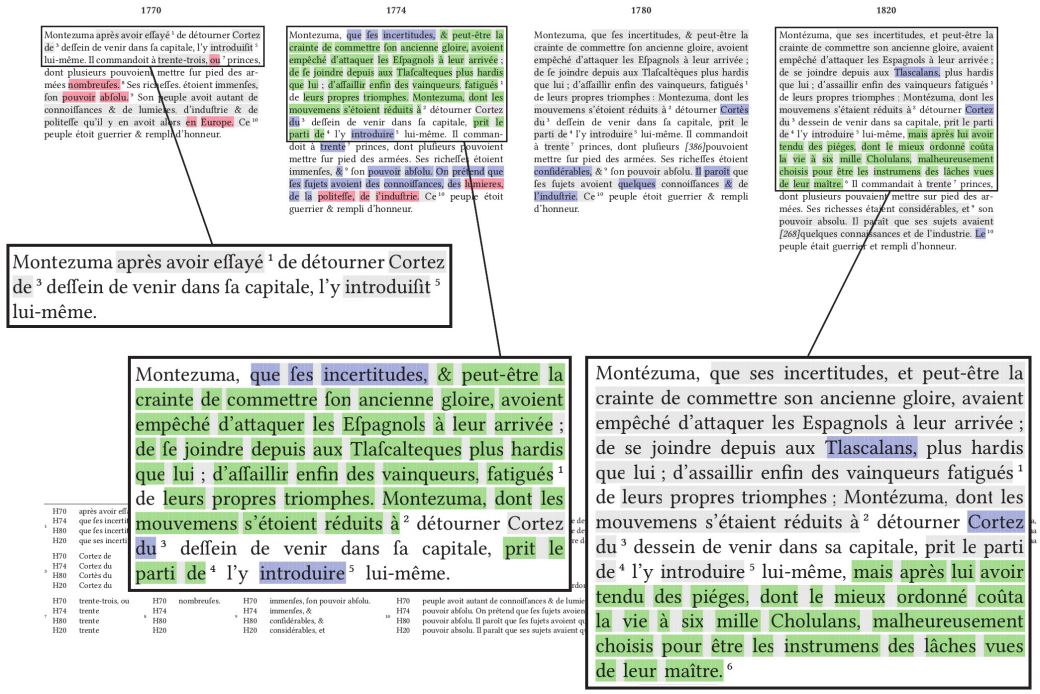


Abbildung 2.1: Ein Abschnitt der *Histoire des deux Indes* zur Begegnung von Montezuma II. und Hernán Cortés in den vier Textfassungen. Der ursprüngliche Satz der Erstausgabe wurde in der Fassung von 1774 inhaltlich stark erweitert und in der Fassung von 1820 erneut ergänzt.

terschiede, das heißt Schreibweisen, die innerhalb einer Textfassung konstant, zwischen den Fassungen aber unterschiedlich sind. Dazu zählen für die *Histoire des deux Indes* unter anderem Leerzeichen vor und nach Interpunktionen, das ausgeschriebene („et“) und kaufmännische Und („&“), die Verwendung des langen s („ſ“)[1] in den ersten drei Fassungen und eine Lautverschiebung bei Verbendungen im Imparfait.

Im Französischen kommt eine weitere Problematik hinzu. Ist ein Wort durch einen Zeilenumbruch im Druckbild in zwei Teile getrennt, kann nicht automatisch von einem Trennstrich zwischen diesen ausgegangen werden. Denn Bindestriche kommen

[1] Das kleine runde s wurde früher in einigen Sprachen wie Französisch und Deutsch nur am Wortende genutzt. Am Anfang und innerhalb des Wortes kam hingegen das lange s (ſ) zum Einsatz. Die Besonderheit der Schrift findet sich analog im Griechischen mit dem kleinen Sigma (am Wortende ς statt σ).

im Französischen sehr häufig vor, beispielsweise in Zahlen wie quatre-vingt-dix. Entsprechend können die Wortteile nicht einfach zusammen geschoben werden, sondern es muss im konkreten Fall geprüft werden, ob ein Trenn- oder Bindestrich vorliegt. Andernfalls kann es zu falschen Textvarianten allein aufgrund des unterschiedlichen Zeilenfalls der Textfassungen kommen.

2.1.2 Keter Shem Ṭov

Bei dem Werk *Keter Shem Ṭov* (Krone des guten Namens) handelt es sich um ein in Hebräisch verfasstes, kabbalistisches Traktat. Es gilt als einer der „wichtigsten Einführungstexte in die esoterischen Lehren der jüdischen Kabbala“ und verbindet zwei mystische Traditionen zu den zehn Manifestationen Gottes und des Tetragramms – dem vierbuchstabigen Gottesnamen – miteinander, wie in [94] beschrieben. Als mutmaßlicher Autor gilt Abraham ben Axelrad von Köln, der *Keter Shem Ṭov* im 13. Jahrhundert verfasst haben soll. Der Text wurde in verschiedenen Regionen Europas sowohl von jüdischen als auch christlichen Kabbalist:innen rezipiert und ist in etwa hundert unterschiedlichen Textfassungen bezeugt, während das Original selbst nicht überliefert ist.
Keter Shem Ṭov ist der zentrale Forschungsgegenstand des von der Deutschen Forschungsgemeinschaft (DFG) geförderten, interdisziplinären Projekts *Synoptische Edition des kabbalistischen Traktats Keter Shem Ṭov mit englischer Übersetzung, Stellenkommentar und rezeptionsgeschichtlichen Studien* [3, 99]. Ziel ist eine kritische Edition des Traktats, die als Print- sowie als digitale Edition erscheinen soll. Dabei ist eine Spaltensynopse der wichtigsten Fassungen angedacht, die neben dem kritischen Apparat samt Stellenkommentaren auch eine englische Übersetzung beinhalten soll. Die Transkription, der Abgleich und die Darstellung so vieler Textfassungen stellen eine besondere Herausforderung für ein Editionsvorhaben dar. Zudem sind neben der Anzahl auch die Unterschiede zwischen den Textfassungen vergleichsweise groß, was auf die lange und komplexe Überlieferungsgeschichte, aber auch auf die Textgattung zurückzuführen ist: So kann das Werk als eine Menge veränderlicher Bausteine angesehen werden, welche die jeweiligen Autor:innen nach ihrer Interpretation überarbeiten [100]. Das äußert sich in verändertem Inhalt, Umfang und Struktur zwischen den Textfassungen. Und auch wenn zwei Textfassungen in etwa die gleichen inhaltlichen Abschnitte aufweisen, ist ihre Reihenfolge mitunter unterschiedlich. Diese starke Varianz findet sich ebenso innerhalb der Abschnitte.
Zwischen den Textfassungen unterscheidet sich zudem die Verwendung von Ab-

kürzungen, beispielsweise für den Gottesnamen beziehungsweise das göttliche Tetragramm, welches in vielen unterschiedlichen Variationen auftritt. Darüber hinaus beinhaltet *Keter Shem Ṭov* viele Bibelzitate, die je nach Textfassung entweder vollständig oder aber nur mit einigen Worten anzitiert werden.
In diesem Editionsprojekt stellt die Sprache des Werks eine weitere Herausforderung dar. Hebräisch wird aufgrund des eigenen Alphabets und der Schreibrichtung von rechts nach links nur von wenigen digitalen Werkzeugen unterstützt (siehe Abschnitt 3.2.6). Darüber hinaus liegt *Keter Shem Ṭov* nicht in modernem Hebräisch vor, sondern wurde in verschiedenen frühen Sprachstufen verfasst.

2.1.3 Kalīla and Dimna

Das Werk *Kalīla and Dimna* ist laut [43] „eines der am meisten verbreiteten und einflussreichsten Bücher in der Geschichte der Menschheit. Als eine Sammlung von Fabeln, die politische Weisheit lehren, überwand es Sprachen, Kulturen und Religionen“[2]. Es weist eine lange und komplexe Überlieferungsgeschichte mit Textfassungen in etwa 40 Sprachen auf. Der Ursprung von *Kalīla and Dimna* lässt sich bis in verschiedenen Quellen aus dem Sanskrit des dritten Jahrhunderts zurückverfolgen, die ab dem sechsten Jahrhundert ins Mittelpersische und schließlich im achten Jahrhundert ins Arabische übersetzt wurden. Arabisch gilt dabei als „lingua franca des Nahen Ostens“ [43] und ist von zentraler Bedeutung für die weitere Verbreitung des Werks in vielen Regionen des Orients, Europas und Asiens [55]. Die oft anonymen Übersetzer:innen und Kopist:innen arbeiteten das Werk mitunter stark um, sodass eine Vielzahl, insbesondere arabischer Überlieferungen mit teils sehr großen Unterschieden existiert. Die späteren arabischen Fassungen gliedern sich dabei häufig in siebzehn oder achtzehn Kapitel mit den einzelnen Fabeln [54].
Bisher existierte keine kritische Edition zu diesem bedeutenden Werk. Mit dem Pilotprojekt *Kalīla wa-Dimna – Wisdom encoded* [43] an der Freien Universität Berlin wurde zunächst die technische Machbarkeit einer solchen Edition evaluiert, bevor sie schließlich im Rahmen des durch den European Research Council geförderten Projekts *The Arabic Anonymous in a World Classic* [53, 126], kurz *AnonymClassic*, umgesetzt wird. Der Fokus des Editionsvorhabens liegt dabei auf den zwischen dem 13. und 19. Jahrhundert entstandenen arabischen Textfassungen von *Kalīla and*

[2] Aus dem Englischen: „Kalīla and Dimna is one of the most widespread and influential books in the history of humanity. A collection of tales teaching political wisdom, it transcended languages, cultures and religions.“ [43].

Dimna[3], auch wenn noch andere Fassungen im Rahmen des Projekts editiert werden. Ziel ist eine kritische Edition mit synoptischer Gegenüberstellung ausgewählter Textfassungen, welche die Bandbreite der knapp 100 verfügbaren Überlieferungen abdeckt. Dabei sollen auch in diesem Editionsvorhaben die ausgewählten Textfassungen in der Synopse gleichrangig behandelt werden, statt auf eine Leitquelle Bezug zu nehmen.

Die arabischen Textfassungen von *Kalīla and Dimna* als zentraler Untersuchungsgegenstand bergen Herausforderungen für das Editionsvorhaben. In seiner komplexen Überlieferungsgeschichte wurde das Werk von den unterschiedlichen Kopist:innen stark umgearbeitet, sodass es eine Vielzahl von Überlieferungen mit teils sehr großen Unterschieden gibt. Die Filiation ist dabei vielfach ungeklärt. Folglich müssen zunächst die Zusammenhänge und Beziehungen der Textfassungen untersucht werden, bevor repräsentative Fassungen für die Edition ausgewählt werden können. Dabei wurden zwei Phänomene identifiziert, welche die Ähnlichkeit zwischen vielen der Textfassungen zusammenfassen: „Continua“ – der strukturelle Aufbau bleibt recht konstant, während es innerhalb der Textsegmente mitunter zu komplexen Textvarianten kommt – und „Cross-Copying“ – Kopist:innen haben mehrere Manuskripte als Vorlage genutzt und dabei einzelne Teilstücke verschiedener Textfassungen übernommen [54, S.259].

Innerhalb der einzelnen Fabeln von *Kalīla and Dimna* gibt es erzählerische Einheiten, die sich trotz großer Varianz zwischen den Textfassungen, zumindest manuell, wiederfinden lassen. Sie sind hilfreich zur Orientierung beim Abgleich der Textfassungen und dem Erkennen struktureller Unterschiede, wie den vorkommenden Umstellungen in der Erzählreihenfolge.

Illustrationen kommen in gut einem Drittel der Manuskripte vor und sind für das Editionsprojekt ebenfalls wichtig, da sie unter anderem bei der Datierung, Filiation und literarischen Analyse der Textfassungen helfen [54, S.151].

Die arabische Sprache bringt weitere Herausforderungen mit sich, wie sie teilweise auch im Hebräischen auftreten und entsprechend schon beim Abschnitt zu *Keter Shem Ṭov* erwähnt wurden: das Nicht-Lateinische Alphabet und die Schreibrichtung. Darüber hinaus besitzen die meisten Wörter im Arabischen eine aus drei oder vier Konsonanten bestehende Wortwurzel, welche die grundlegende Bedeutung eines Wortes vorgibt. Zusätzliche Vokale, Prä- und Suffixe dienen zur Bildung der eigentlichen Wortform. In den Textfassungen werden dabei häufig nur die Konso-

[3] In Bezug auf die arabischen Textfassungen spricht das Projekt von *Kalīla wa-Dimna* [54, S.243].

nanten oder langen Vokale der Wörter explizit ausgeschrieben, während kurze Vokale meist wegfallen. Sie werden mitunter allerdings durch Diakritika angedeutet. Die genaue Bedeutung der Worte ergibt sich dadurch erst aus dem Kontext, was den Abgleich der Textfassungen zusätzlich erschwert. Zudem werden Wörter teilweise zusammengezogen, beispielsweise Substantive mit dem dazugehörigen Artikel.

2.1.4 Martín de Azpilcuetas Manual de confesores y penitentes

Martín de Azpilcueta (* 1491 in Barásoain, Navarra; † 1586 in Rom), ein Gelehrter des kirchlichen Rechts und Professor an den Universitäten von Salamanca sowie Coimbra, formte im 16. Jahrhundert das auf einen anonymen Franziskanermönch zurück gehende *Manual de confesores y penitentes* – also ein *Handbuch für Beichtväter und Büßer*. Das umfangreiche Regelwerk zur Verfahrensweise und Bewertung von Beichten gilt insbesondere als Antwort auf Fragen, die sich zu jener Zeit im Zusammenhang mit der Entdeckung der Neuen Welt stellten [18, S.2] und hat „in hohem Maße zur frühneuzeitlichen Rechtsbildung auf globaler Ebene beigetragen“[4]. Das Werk weist eine komplexe Druckgeschichte mit verschiedenen Ausgaben, Nachdrucken und Übersetzungen auf, welche zum Teil auf Azpilcueta selbst zurückgehen. So wurde das Handbuch ursprünglich für das portugiesische Reich geschrieben, später dann für das spanische Einflussgebiet angepasst und letztendlich unter der Leitung des Papstes an die gesamte christliche Welt gerichtet [18, S.18].
Das *Manual de confesores y penitentes* ist Gegenstand des Forschungs- und Editionsprojekts *HyperAzpilcueta. Visualizing the Instability of Early Modern Normative Knowledge* [19] sowie dessen Vorläufer *Martín de Azpilcueta's Manual for Confessors and the Phenomenon of Epitomisation* [20] am Max-Planck-Institut für Rechtsgeschichte und Rechtstheorie in Frankfurt. Dabei liegt der Fokus auf den von Azpilcueta selbst erstellten beziehungsweise betreuten Überarbeitungen. Ausgehend von der 1549 auf Portugiesisch erschienenen Textfassung werden die 1552 in Coimbra ebenfalls auf Portugiesisch, die 1556 in Salamanca auf Spanisch und die 1573 in Rom auf Lateinisch erschienenen Fassungen untersucht [19]. Hinzu kommt eine weitere spanische Fassung von 1553 (Coimbra), die eine direkte Übersetzung der Fassung von 1552 ohne große inhaltliche Eingriffe ist [148]. Ziel des Projekts ist eine digitale Edition mit der synoptischen Gegenüberstellung der Fassungen, wel-

[4] Aus dem Englischen: „Having largely contributed to early modern legal literacy on a global scale, this book [...]“ [19].

che die Überarbeitungen des Textes durch den Autor nachvollziehbar machen und einordnen soll. Damit gehen Forschungsfragen zu inhaltlichen Änderungen durch historische Entwicklungen und äußere Einflüsse auf den Autor einher. Die Edition soll den Leser:innen zudem Faksimile bereitstellen, das heißt hoch auflösende Bilder der originalen Drucke.
Über die vier beziehungsweise fünf Textfassungen hinweg gibt es große Überarbeitungen des Inhalts, aber auch der Struktur und des Layouts. Aus dem kleinen, ursprünglichen Handbuch auf Portugiesisch wurde so schrittweise ein umfangreiches lateinisches Werk, das sich schließlich mit einem unterschiedlichen Verwendungszweck an andere Leser:innen richtet als das Original [18, S.3]. Diese „Instabilität“ ist zum einen auf die Intention des Autors, zum anderen aber auch auf technische oder ökonomische Gründe beim Druck selbst zurückzuführen [19]. Da die Übersetzung aus dem Portugiesischen (1552) ins Spanische (1553) sehr originalgetreu ist, geschehen die radikalen Texteingriffe zunächst nur innerhalb der selben Sprache, das heißt innerhalb der portugiesischen Fassungen von 1549 und 1552 sowie innerhalb der spanischen Fassungen von 1553 und 1556 [18, S.17]. Das ändert sich mit der Übersetzung in die lateinische Fassung (1573), in welcher auch der Inhalt selbst durch Überarbeitung, Hinzufügung, Streichung und Verschiebung ganzer Abschnitte stark verändert wird [18, S.17].
Das Editionsvorhaben muss mit verschiedenen Herausforderungen gleichzeitig umgehen. Erstens wurde der Inhalt des Werkes stark überarbeitet und erweitert sowie die Struktur und in Teilen das Layout verändert. Zweitens liegt das Werk in unterschiedlichen Sprachen vor, wobei umfangreiche Textvarianten teilweise auch während der Übersetzung (vor allem ins Lateinische) entstanden. Demnach ist ein Textvergleich auch über Sprachgrenzen hinweg unerlässlich. Drittens hat der Text sprachliche Besonderheiten, wie ein historisches, teils domänenspezifisches Vokabular sowie eine historische Orthographie (in gleich drei Sprachen), eine inkonsistente Interpunktion und eine große Anzahl von Abkürzungen [148]. Hinzu kommen Elemente im Layout, wie Fußnoten und Marginalien, die wichtige Informationen enthalten und im Rahmen der digitalen Edition stets mitgedacht werden müssen.

2.1.5 Hannah Arendt. Kritische Gesamtausgabe

Hannah Arendt (* 1906 in Linden (Hannover); † 1975 in New York) war eine bedeutende Publizistin des 20. Jahrhunderts, die mit ihren Werken zur politischen Theorie die Geschehnisse des 20. Jahrhunderts kritisch begleitet und einordnet. Als Jüdin

in ihrem Geburtsland Deutschland verfolgt, emigrierte sie 1933 nach Frankreich und 1941 schließlich in die USA. Fast all ihre Werke hat sie in mehreren Sprachen verfasst, insbesondere Deutsch und Englisch, auch begründet mit ihrem jeweiligen Aufenthaltsort.

Das von der Deutschen Forschungsgemeinschaft (DFG) ab 2020 geförderte Langzeitprojekt *Hannah Arendt. Kritische Gesamtausgabe* [1, 81] an der Freien Universität Berlin widmet sich Hannah Arendts Schriften mit dem Ziel einer hybriden Edition. Während die Druckfassungen der Edition konstituierte Texte präsentieren, soll das entstehende Webportal [58] die Textvarianten nicht nur als diplomatische Auszeichnungen darstellen, sondern auch eine synoptische Gegenüberstellung der von den Nutzer:innen ausgewählten Textfassungen ermöglichen. Neben der umfangreichen Dokumentation der Textgenese und detaillierten Stellenkommentaren sollen beim Webportal zudem Faksimile sowie diplomatische Transkriptionen in XML bereitgestellt werden. Beide Medien sollen der Mehrsprachigkeit Hannah Arendts und ihrer Werke gerecht werden. Die rund 21.000 zu edierenden Seiten umfassen die veröffentlichten Werke sowie viele unveröffentlichte Dokumente aus dem Nachlass der Autorin, das heißt insbesondere auch handschriftliche Notizen und Umarbeitungen. Sie werden im Rahmen des Editionsvorhabens in siebzehn Themenkomplexe eingeteilt und nach dem Editionsplan des Projekts im Zeitraum von 2018 bis 2031 bearbeitet.

Auch dieses Editionsprojekt bringt spezifische Herausforderungen mit sich. Zum Ersten handelt es sich um mehrere, teils sehr lange Werke mit mehreren Textfassungen. Entsprechend gibt es sehr viel Text, der kollationiert werden muss. Zum Zweiten weisen die Werke zahlenmäßig viele und umfangreiche Umarbeitungen durch die Autorin auf, was insbesondere dadurch verstärkt wird, dass auch ihr Nachlass mit den mitunter frühen Fassungen der Texte Teil des Editionsvorhabens ist. Hinzu kommt ein besonderes Augenmerk auf handschriftliche Notizen, wie Ergänzungen, Streichungen oder Korrekturen, die im Zuge des Projekts ebenfalls transkribiert, verarbeitet und dargestellt werden müssen, um einen „Blick in die Denk- und Schreibprozesse der Autorin“ zu ermöglichen [58]. Zum Dritten werden beide Aspekte durch die Mehrsprachigkeit der Textfassungen verschärft. Zwar handelt es sich um im Natural Language Processing viel beachtete Sprachen in modernen Sprachstufen – hauptsächlich Deutsch und Englisch, teils auch Französisch und Jiddisch –, doch hat Hannah Arendt ihre Werke nicht direkt übersetzt, sondern beim Wechsel der Sprache umgearbeitet. Der teils philosophische Inhalt, bei dem einzel-

ne Wörter sehr viel Bedeutung haben können, macht den Abgleich der Fassungen dabei besonders herausfordernd.

2.2 Problemklassen

Die oben vorgestellten Werke und Editionsvorhaben unterscheiden sich in vielen Aspekten, doch lassen sich auch Gemeinsamkeiten auf mehreren Ebenen erkennen. Der folgende Abschnitt arbeitet diese mittels dreier Fragestellungen heraus: Erstens, wie ist die Beschaffenheit der zu untersuchenden Textfassungen? Zweitens, wie ist die Beschaffenheit der Varianten zwischen den Textfassungen? Und drittens, welches Ziel wird mit der Untersuchung der Textvarianten verfolgt? Die Antworten auf diese drei Fragen liefern konkrete Problemklassen für die (automatische) Kollationierung, auf deren Grundlage Anforderungen für digitale Werkzeuge in dieser Domäne abgeleitet werden können.

2.2.1 Beschaffenheit der Textfassungen

Die in einem Editionsprojekt betrachteten Textfassungen können sehr divers sein, wie die in Abschnitt 2.1 aufgeführten Beispiele zeigen. Das gilt sowohl innerhalb eines Projekts, als auch deutlich verstärkt über Projektgrenzen hinweg. Die folgenden drei Attribute zur Beschaffenheit von Textfassungen – Anzahl, Umfang und Sprache – helfen bei der Systematisierung.

Die **Anzahl** der zu untersuchenden Textfassungen ist ein wesentliches Kriterium zur Einordnung. Bereits der Vergleich zweier unterschiedlicher Fassungen kann eine in Bezug auf die Methodik und den zu betreibenden Aufwand komplexe Aufgabe darstellen. Weitere zu vergleichende Textfassungen verschärfen die bestehenden Probleme nicht nur, sondern werfen weitere auf. Dazu zählt insbesondere der prinzipielle Ablauf des Vergleichs: Werden alle Fassungen mit allen verglichen oder gibt es einen Basistext, bezüglich dessen die Textvarianten bestimmt werden?
Letzteres wird als Leitquellenprinzip bezeichnet und findet breite Anwendung bei gedruckten Editionen, da es die Anzahl der Textvarianten verringert beziehungsweise verkürzt und so beim Druck sehr viel Platz spart.
Dem diametral gegenüber steht die gleichrangige Kollationierung, bei der es keine festgelegte Ordnung der Textfassungen gibt und entsprechend alle paarweise mitein-

ander verglichen werden. Das ist aus editorischer Sicht sinnvoll, wenn die Fassungen als gleichwertig betrachtet und präsentiert werden sollen, wie zum Beispiel bei der *Histoire des deux Indes* oder *Keter Shem Ṭov* oder aber es keine (bekannte) historische Chronologie der Textfassungen gibt, wie das häufig bei frühen handschriftlichen Manuskripten der Fall ist (siehe beispielsweise *Kalīla and Dimna*). Dieses Vergleichsprinzip wird erst im digitalen Medium praktikabel, wo die Beschränkungen von Printmedien (Seitenformat und -anzahl, Farbgestaltung, Unveränderlichkeit) größtenteils aufgehoben werden.

Zwischen Leitquellenprinzip und gleichrangigem Vergleich gibt es einen Mittelweg, bei dem nur benachbarte Textfassungen entsprechend einer vorgegebenen, gegebenenfalls frei gewählten Chronologie miteinander verglichen werden, was den Überarbeitungsprozess eines Werks intuitiv abbildet und so verständlich macht. Ein Beispiel hierfür ist das *Manual de confesores y penitentes*.

Eine steigende Anzahl von Textfassungen führt darüber hinaus zur Verschärfung bestehender Probleme in Bezug auf die Berechnung der Kollationierung. So macht es einen algorithmischen Unterschied, ob 5, 10, 20 oder gar 100 Textfassungen miteinander verglichen werden, vor allem wenn die Methoden in einer interaktiven Arbeitsumgebung ausgeführt werden sollen. Schließlich hat die Anzahl der Textfassungen auch Einfluss auf die Darstellung und Navigation innerhalb einer digitalen Edition. Während manche Darstellungsformen nur für wenige Textfassungen gleichzeitig geeignet sind, entfalten andere Visualisierungen ihr Potenzial erst ab einer gewissen Anzahl.

Der **Umfang** der einzelnen Textfassungen ist ein zweites wichtiges Kriterium. Bei den oben aufgeführten Editionsprojekten reicht das Spektrum von kurzen Manuskripten (*Keter Shem Ṭov*) oder einzelnen Abschnitten (die einzelnen Fabeln von *Kalīla and Dimna*) bis hin zu ganzen Büchern, die am Stück verglichen und dargestellt werden sollen (*Histoire des deux Index*).

Letzteres resultiert in besonderen Herausforderungen für Editor:innen und die verwendete Technik, was insbesondere die Algorithmen zur effizienten Berechnung der Unterschiede in den Textfassungen, aber auch die übersichtliche Darstellung betrifft. Zudem erschweren große Textmengen die Navigation innerhalb einer Arbeitsumgebung oder der digitalen Edition.

Die **Sprache** sowie die Sprachstufen der Textfassungen ist ein drittes Kriterium zur

Einordnung eines Editionsvorhabens. Die oben aufgeführten Werke und Projekte umfassen eine ganze Reihe von Sprachen verschiedener historischer Zeiträume. Damit einher gehen verschiedene sprachspezifische Phänomene und Probleme. Eine Sprache bringt mitunter ein selten verwendetes Alphabet, eigene Sonderzeichen oder Diakritika mit sich, die von einem digitalen Werkzeug verarbeitet und dargestellt werden müssen.
Nichtlateinische Alphabete sind auch häufig gepaart mit einer anderen Schreibrichtung als Links-nach-Rechts, die das System entsprechend berücksichtigen muss.
Zwischen den untersuchten Textfassungen kann sich die Sprache zudem weiterentwickeln, beispielsweise durch die Anpassung der Orthographie, was wiederum zu einer Vielzahl identischer Textvarianten führen kann, die über den ganzen Text verteilt sind und so die eigentliche Untersuchung stören. Das zeigt das Beispiel der *Histoire des deux Indes* bei der in der Textfassung von 1820 das runde (s) das lange s (ſ) ersetzt und Verb-Endungen der Vergangenheitsform Imparfait abweichen.
Im Extremfall gibt es verschiedensprachige Textfassungen, die miteinander verglichen werden sollen, wie bei den Werken von Hannah Arendt oder dem *Manual de confesores*. Bei letzterem kommt dem Vergleich eine übergreifende Struktur der Textfassungen zu Gute, welche in manchen Genres, wie Rechts- oder religiösen Texten, vorzufinden ist.

2.2.2 Beschaffenheit der Textvarianten

Die Unterschiede zwischen Textfassungen können verschiedene Hintergründe und Ausprägungen haben. Die vorgestellten Werke und Editionsvorhaben in Abschnitt 2.1 verdeutlichen, dass nicht immer alle Arten von Textvarianten auftreten oder im Forschungsinteresse eines speziellen Projekts liegen beziehungsweise ganz allgemein bei der Anwendung automatischer Kollationierungssoftware von Relevanz sind. Im Folgenden wird eine grobe Kategorisierung von Textvarianten vorgenommen, welche die spätere Einordnung im Rahmen dieser Dissertation erleichtert. Die Kategorien sind dabei nicht diametral zueinander, sondern überlagern sich in vielen Fällen.

Eine Ursache von Textvarianten sind kleine **Tipp- oder Rechtschreibfehler**, die oft nur einzelne Buchstaben betreffen. Werden sie im Laufe einer Überarbeitung korrigiert, so erzeugen sie einen Unterschied zwischen den Textfassungen. Sie treten in den meisten Editionsprojekten auf, stammen mitunter aber nicht immer von den

eigentlichen Autor:innen, sondern können durch die Weiterverarbeitung entstehen. So sind sie beispielsweise bei der entstehenden Edition der Werke von Sir Arthur Doyle von Interesse, bei der verschiedene Druckauflagen der Texte auf kleinste Abweichungen hin miteinander verglichen werden [6].

Eine zweite Kategorie umfasst die Änderung von einzelnen **Sonderzeichen** abseits von Buchstaben und Zahlen, wie zum Beispiel Interpunktionen und Diakritika. Sie können mitunter große Auswirkungen für die Bedeutung einer Phrase haben, tun dies in der Regel aber selten, sodass sie bei vielen Anwendungsszenarien ignoriert werden können und sollten.

Änderungen einzelner Buchstaben oder Sonderzeichen können auch systematischer Natur sein und einem bestimmten (wiederkehrenden) **Muster** folgen, beispielsweise in Folge einer Rechtschreibreform – siehe langes s oder Imparfait bei der *Histoire des deux Indes* – oder bei wiederkehrenden Textvarianten, die auf Eigenheiten des Schreibstils der Autor:innen zurückzuführen sind. Ein weiteres Beispiel sind Abkürzungen, die wie im Falle von *Keter Shem Ṭov* – hier werden Bibelzitate manchmal ausgeschrieben und manchmal nur anzitiert – auch ganze Phrasen umfassen können.

Die nächsten drei Kategorien gehen mit einer inhaltlichen Überarbeitung einher. Dabei ist es wesentlich, in welchem Umfang diese Textvarianten auftreten: Finden sie auf Wortebene statt oder sind längere Textsegmente, wie Sätze oder Absätze betroffen?

Zum einen sind das **Substitutionen**, das heißt Ersetzungen oder Umformulierungen am Platz, die sich in allen oben aufgeführten Editionsprojekten finden. Zum anderen werden Texte aber auch durch **Ergänzungen**, das heißt das Einfügen neuen Inhalts, und **Streichungen**, also das Entfernen bestehenden Inhalts, überarbeitet. Ob es sich um Ergänzungen oder Streichungen handelt ist dabei abhängig von der gewählten Reihenfolge beim Textvergleich, wenn keine Entstehungsreihenfolge gegeben ist. Bei der *Histoire des deux Indes* sowie bei *Kalīla and Dimna* findet sich beides im großen Umfang auch auf Absatzebene.

Ergänzend zu diesen drei Kategorien kommen Umstrukturierungen in Form von **Transpositionen** hinzu. Dabei handelt es sich um Vertauschungen des Inhalts, die sowohl großflächig für längere Textsegmente als auch auf Wortebene in Form von Satzumstellungen auftreten. Die Textvarianten von *Keter Shem Ṭov* weisen vielfach

Transpositionen auf, wobei einige Manuskripte gar die inhaltliche Abfolge des halben Werks vertauschen.
Diese vier Operationen werden oft gemeinsam verwendet, wenn Werke durch ihre Autor:innen stark überarbeitet werden. So entstehen teils massive Varianzen und Längenunterschiede zwischen den Textfassungen.

Trotz starker Unterschiede auf der Textoberfläche kann eine Textvariante den Inhalt auch unberührt lassen. So können zwei Phrasen eine identische Aussage treffen ohne auch nur ein gemeinsames Wort zu besitzen. Daher ist eine weitere Kategorie sinnvoll, die aufzeigt, inwiefern eine Textvariante die **Semantik** berührt. Eine Fokussierung auf die semantische Ähnlichkeit beim Textvergleich kann sehr hilfreich sein, wenn inhaltliche Unterschiede untersucht werden sollen oder der Vergleich über Sprachgrenzen hinweg stattfindet. Übersetzungen bringen ohnehin eine gewisse Unschärfe bei der Wortwahl mit sich.

Im weiteren Sinne können Textvarianten nicht nur auf der Textoberfläche auftreten, sondern auch in Bezug auf die **Typografie**. Manche Textfassungen nutzen beispielsweise Kapitälchen oder Ligaturen, die in anderen nicht auftreten und so Textvarianten erzeugen. Solche typografischen Unterschiede können sprachhistorisch relevant sein. Ein Beispiel [5] hierfür findet sich in Albrecht von Hallers *Versuch Schweizerischer Gedichte(n)*, welches in insgesamt elf Auflagen erschien. Abbildung 2.2 zeigt einen Ausschnitt aus dem Gedicht „Uber die Ehre. Als Herr *D.* Giller den *Doctor*-Hut annahme.“ in der Erstausgabe von 1732. Das Wort *Doctor* beziehungsweise

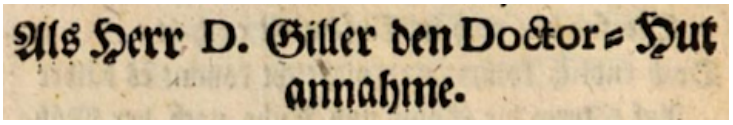


Abbildung 2.2: Auszug aus *Versuch Schweizerischer Gedichten* von Albrecht von Haller in der ersten Auflage von 1932 (Seite 34, `https://books.google.de/books?id=2PM9AAAAcAAJ&pg=PA34`).

D. wird hier als lateinisches Fremdwort verstanden und entprechend als fremdsprachiger Einschub in Antiqua gesetzt, während der restliche Text als Fraktur gesetzt wurde. Dies ändert sich in späteren Auflagen, wie in Abbildung 2.3 zu sehen: Das

[5] Zugetragen in privater Korrespondenz von Frank Zöllner (Friedrich-Schiller-Universität Jena).

einstige Fremdwort ist in den allgemeinen Sprachgebrauch übergegangen und wird nun ebenfalls in Fraktur gesetzt. Zwischen den Textfassungen findet also eine semantische Verschiebung des Worts *Doctor* (sowie weiterer Wörter) statt, die allein an der Typografie aber nicht an der Textoberfläche erkennbar ist.

Abbildung 2.3: Auszug aus *Versuch Schweizerischer Gedichte* von Albrecht von Haller in der neunten Auflage von 1962 (Seite 9, `https://books.google.de/books?id=gWZAAQAAMAAJ&pg=PA9`).

2.2.3 Zielsetzung der Kollationierung

So wie es Unterschiede zwischen den betrachteten Werken und Editionsvorhaben gibt, so unterscheiden sich auch die konkreten Ziele, die mit dem Abgleich unterschiedlicher Textfassungen verfolgt werden. Ergänzend zur Einleitung und basierend auf den oben vorgestellten Projekten werden im Folgenden vier konkrete Anwendungsszenarien für die automatische Kollationierung diskutiert. Dabei stellt sich insbesondere die Frage, welche Arten von Textvarianten für die jeweilige Zielstellung relevant sind und welche gegebenenfalls mithilfe von geeigneter Normalisierung und Filterung ignoriert werden können.

Eine notwendige Grundlage für die automatische (digitale) Kollationierung sind maschinenlesbare Versionen der Textfassungen. Gleichzeitig kann die Kollationierung wiederum beim Auffinden von Fehlern in den grundlegenden Dateien helfen, da diese als Textvarianten hervorgehoben werden.
Der Prozess der digitalen Erfassung der Textfassungen heißt Transkription, wobei es im Umfeld der digitalen Editionsphilologie zwei gängige Verfahren gibt. Zum einen ist das die Erfassung durch fach- oder gar sprachfremde Personen, die zeichenweise ein Faksimile, oft eine Fotografie oder einen Scan der Textfassung, abtippen. Um die Korrektheit zu erhöhen, wird dieser Vorgang oft unabhängig voneinander von zwei Personen durchgeführt und bei Unterschieden nochmals genauer geprüft. Für dieses sogenannte Double-Keying-Verfahren gibt es spezialisierte Firmen, die Genauigkeiten bis zu 99.997% erreichen [46, S.11]. In einer Studie des Deutschen

Textarchivs [57] wird die hohe Genauigkeit von 99.97%, das heißt rund einen Fehler alle drei Seiten, für historische Texte bis ins 18. Jahrhundert bestätigt. Die verbleibenden Fehler beziehen sich demnach häufig auf optisch ähnliche Buchstaben, wie f und ſ (das lange s) oder u und n, insbesondere wenn unterschiedliche Schriftarten hinzukommen.

Kommerzielles Double-Keying steht vielen Projekten allerdings nicht zur Verfügung. Eine gängige Alternative ist ein semi-automatischer Ansatz mittels bildbasierter Texterkennung (OCR – Optical Character Recognition, beziehungsweise HCR – Handwritten Character Recognition) mit anschließenden Korrekturgängen. Die ohnehin geringere Genauigkeit entsprechender Software im Vergleich mit Double-Keying sinkt dabei deutlich bei der Anwendung auf sonst selten verwendete Sprachen oder Sprachstufen sowie bei untrainierten Schriftarten und vor allem handschriftlichen Manuskripten.

Gerade zu Beginn eines Editionsprojekts kann die Verbesserung der Transkriptionen daher die wesentliche Motivation für die automatische Kollationierung der Textfassungen sein. Da das Hauptaugenmerk auf Tippfehlern liegt, ist eine Normalisierung wie bei den folgenden Anwendungsszenarien nicht hilfreich, sondern im Gegenteil sogar kontraproduktiv.

Das zweite Anwendungsszenario berührt Editionen, die als substanzielle Grundlage für die weitere Forschung konzipiert werden, indem der überlieferte Textbestand möglichst exakt dokumentiert wird. Zur Vollständigkeit solcher Editionen gehören auch die Varianten zwischen den verschiedenen Fassungen des Textes. Hauptaugenmerk sind dabei die Unterschiede, welche für die Leserschaft in einem kritischen Apparat aufbereitet werden. Da auch Varianten ohne Bedeutungsverschiebung relevant sind, ist auch hier in der Regel keine umfassende Filterung angebracht.

Teils erfahren systematische Unterschiede zwischen den Textfassungen aber eine Sonderbehandlung: Statt alle Vorkommen aufzulisten, wird solch eine Variante verbal zusammengefasst und mit einigen Belegstellen exemplarisch veranschaulicht. In der Darstellung des Fließtextes werden sie dann gegebenenfalls normalisiert. Dies kann zum Beispiel eine Änderung der gängigen Orthografie betreffen oder einen konkreten Bezug zum Schreibstil der Autor:innen aufweisen. Hinzu kommen zudem Normierungen für die Lesefassung, beispielsweise die Anpassung an eine moderne Sprachstufe.

Beim dritten Anwendungsszenario werden die Gemeinsamkeiten und Unterschiede zwischen den Textfassungen zur Erforschung von Traditionslinien verwendet. Das beinhaltet zum einen das Aufzeigen des Abstammungsverhältnisses der Textfassungen (ihre Filiation) und darauf aufbauend zum anderen das Nachzeichnen der Entstehungs- und Änderungshistorie innerhalb der Traditionslinien (die Textgenese). Die Filiation ist gerade bei frühzeitlichen Manuskripten, die über handschriftliche Kopien verbreitet wurden, unklar. Ihre Untersuchung liefert nicht nur Rückschlüsse auf die Verbreitung von Wissen, sondern auch die Grundlage zur Rekonstruktion des Urtextes, insofern dieser nicht überliefert wurde.

Für dieses Anwendungsszenario automatischer Kollationierung ist vor allem das Veranschaulichen struktureller Änderungen wichtig sowie das Aufzeigen markanter Stellen, die nur in wenigen Textfassungen vorkommen. Beides zeigt die textuelle Ähnlichkeit zweier Textfassungen in Abgrenzung zu den anderen an und liefert somit Evidenz dafür, dass eine von der anderen abstammen könnte. Einzelne Varianten auf der Textoberfläche können dabei mitunter eine entscheidende Rolle spielen, wenngleich vor allem umfangreiche oder semantische Eingriffe bei der Einordnung helfen. Entsprechend ist eine Normalisierung und Filterung im Einzelfall abzuwägen und sollte optional gehalten werden.

Das vierte Szenario hat schließlich die Textvarianten selbst im Blick. Als Kern historisch kritischer Editionen erlaubt ihre Analyse Rückschlüsse zur Beantwortung historischer Fragestellungen. Dazu gehört auch die Einordnung in den jeweiligen historischen Kontext durch den Abgleich mit weiteren Quellen, um die Distanz zwischen der damaligen Wirklichkeit und dem heutigen Verständnis zu überbrücken (vgl. Sahle in [77, S.239]).

In diesem Anwendungsszenario liegt das Hauptinteresse auf den inhaltlichen Änderungen. Demnach ist der Einsatz von Normalisierungen zur Erleichterung der editorischen Arbeit sinnvoll. Das umfasst insbesondere Filteroptionen, um die für die verfolgten Forschungsfragen irrelevante Textvarianten auszublenden, sprich herauszufiltern. So existieren zwischen den vier Fassungen der *Histoire des deux Index* sehr viele Textvarianten, die auf Unterschiede in der Groß-/Kleinschreibung, bei der Verwendung des langen/runden s oder auch Lautverschiebungen bei Verbendungen im Imparfait zurückgehen. Diese können ignoriert werden, sollen inhaltliche Fragen, beispielsweise zur Entwicklung des Wissensstands des Autors, mittels der Analyse der Textvarianten untersucht werden.

2.3 Anforderungen

Aus den beschriebenen Problemklassen in Bezug auf die Beschaffenheit der Textfassungen und -varianten sowie den Zielsetzungen der Kollationierung leiten sich konkrete Bedarfe und Anforderungen an digitale Werkzeuge für die Editionsphilologie ab. Im Folgenden werden sechs wesentliche Bedarfe beschrieben, auf die in den folgenden Kapiteln Bezug genommen wird. Diese Bedarfe berühren dabei sowohl die Methoden bei der Durchführung des Textvergleichs als auch die digitale Darstellung der gewonnenen Vergleichsergebnisse für die Editor:innen beziehungsweise die spätere Leserschaft. Aus diesen Anforderungen leiten sich wiederum die zentralen Forschungsfragen ab, auf welche die Dissertation eingehen wird. Die sechs Bedarfe sind:

I. Vergleich von mehreren Textfassungen

II. Vergleich von umfangreichen Textfassungen

III. Umgang mit komplexen Textvarianten

IV. Nutzerfreundlichkeit des Systems

V. Interaktivität des Systems

VI. Generizität des Systems

I. Bei der Kollationierung werden verschiedene Textfassungen miteinander verglichen. Dabei ist zum Ersten der Vergleich von mehr als zwei Textfassungen ein wesentlicher Bedarf aller der hier beschriebenen Editionsprojekte.
In Abgrenzung zum klassischen Leitquellenprinzip aus dem Printbereich, bei dem die Textfassungen ausschließlich mit einem ausgewählten Basistext verglichen werden, ist zum Zweiten vielfach ein gleichrangiger Vergleich, das heißt ein paarweiser Vergleich aller Textfassungen miteinander, als Vergleichsprinzip geboten. Dieser Ansatz ist sinnvoll, wenn beispielsweise die Chronologie der Textfassungen nicht bekannt – wie bei vielen handschriftlichen Überlieferungen – und die Auswahl eines Basistextes vor der eigentlichen Textanalyse nicht wirklich möglich ist. Gleiches gilt, wenn, wie im Falle der *Histoire des deux Indes*, jede Textfassung ihr „eigenes Recht“ [23] hat und somit alle Textvarianten forschungsrelevant sind. Spencer und Mowe analysieren in [143] die Nachteile des klassischen Leitquellenprinzips und kommen

zu dem Schluss, dass es insbesondere im Rahmen stemmatologischer Analysen, also der Rekonstruktion des Abstammungsverhältnisses der Textfassungen (Filiation), keine ideale Lösung darstellt[6], da ein entsprechendes Verfahren zum einen die Paarungen ohne Leitquelle nicht berücksichtigt und entsprechend nicht *optimal* kollationiert, und zum anderen korrespondierende Textpassagen nicht direkt rekonstruiert werden können.
Zum Dritten kommt als Anforderung der Umgang mit teils sehr vielen Textfassungen hinzu, beispielsweise in den Editionsprojekten zu *Keter Shem Ṭov* und *Kalīla and Dimna*.
Mit diesen drei Bedarfen bezüglich des Vergleichs von mehreren Textfassungen gehen verschiedene Forschungsfragen für die Informatik einher: Wie lassen sich die Textvarianten samt korrespondierender Textpassagen beim gleichrangigen Vergleich mehrerer Textfassungen algorithmisch effizient berechnen und anschaulich visualisieren? Und wie muss auf dieser Grundlage eine Software gestaltet beziehungsweise erweitert werden, um nicht nur drei oder vier, sondern gleichzeitig dutzende Textfassungen handhaben zu können?

II. Der Vergleich von sehr umfangreichen Textfassungen ist der zweite wesentliche Bedarf, der sich aus den aufgeführten Editionsprojekten herauskristallisiert. Große Textmengen stellen eine Herausforderung für die zugrunde liegenden Algorithmen in Bezug auf die Laufzeiten zur Berechnung der Textvarianten dar; zudem aber auch für die Darstellung und Nutzerführung. Beim Livre VI der *Histoire des deux Indes* beispielsweise, gibt es zwischen den Textfassungen große strukturelle Differenzen auf Absatzebene, zeitgleich müssen Änderungen auf Wortebene betrachtet werden, um philologische Forschungsfragen in Bezug auf die inhaltliche Entwicklung beantworten zu können. So stellt sich für die Informatik die Frage, wie gleichzeitig Distant- und Close-Reading in diesem Forschungsfeld ermöglicht werden kann und das zudem möglichst effizient in Anbetracht großer Textmengen, wie ganzer Bücher im Falle der *Histoire des deux Indes*.

III. Wie in Abschnitt 2.2.2 aufgeführt, gibt es verschiedenartige und teils sehr komplexe Textvarianten, die wiederum auf verschiedenartige Zielsetzungen (siehe Abschnitt 2.2.3) beim Vergleich von Textfassungen treffen. Daraus leiten sich die bei-

[6] Aus dem Englischen: „Choosing only one witness against which to compare all the others is not an ideal solution when we want to construct a stemma.“ [143, S.254].

den Anforderungen an ein entsprechendes digitales Werkzeug ab, zum einen die verschiedenen Arten von Varianten erkennen und darstellen zu können, als auch zum anderen diese bei Bedarf auszublenden. Bei der inhaltlichen Untersuchung der *Histoire des deux Indes* beispielsweise sind Textvarianten rein bestehend aus geänderten Akzenten, dem langen/runden s oder der Lautverschiebung bei Verbendungen im Imparfait nicht nützlich und sollten vom System ignoriert werden können. Daraus ergibt sich die Frage, wie vermeintlich unwichtige Änderungen auf der Textoberfläche automatisch erkannt und herausgerechnet werden können. Ferner stellt sich die Frage, inwieweit die Bedeutungsähnlichkeit von Textvarianten losgelöst von ihrer Textoberfläche erkennbar ist, sodass die Betrachtung auf semantische Änderungen eingeschränkt werden kann. Der Umgang mit großflächigen Umstrukturierungen, das heißt der Transposition von größeren Textsegmenten, ist eine weitere Anforderung und damit zu beantwortende Forschungsfrage. Darüber hinaus besteht bei einigen Editionsvorhaben prinzipiell auch ein Bedarf typografische Textvarianten sowie Textvarianten über Sprachgrenzen hinweg zu analysieren.

IV. Editor:innen sind meist Geisteswissenschaftler:innen aus den philologischen Disziplinen. Als anvisierte Zielgruppe sollte entsprechend keine technische Ausbildung vorausgesetzt werden, um digitale Werkzeuge zum Textvergleich (aus der Domäne der Editionsphilologie) nutzen zu können. So ergibt sich als vierte wesentliche Anforderung eine einfache und intuitive Handhabung solcher Systeme und damit die Frage, wie ein solches System – trotz der komplexen Aufgaben, die damit erfüllt werden sollen – möglichst nutzerfreundlich gestaltet werden kann.

V. Eng verwoben mit der Nutzerfreundlichkeit ist auch der Wunsch nach einer interaktiven Nutzung. Sie wird im Rahmen dieser Dissertation als fünfter wesentlicher Bedarf angesehen. Dazu gehört vor allem die interaktive Wahl der Parameter zum Bestimmen von Textvarianten, wie den zu vergleichenden Textfassungen und ihrer Normalisierung, sodass Editor:innen (sowie die späteren Nutzer:innen der digitalen Edition) je nach Editionsrichtlinien und wissenschaftlichem Interesse entscheiden, aber auch experimentieren können. Dazu muss der Textvergleich ohne lange Wartezeiten neu berechenbar sein, was aufgrund der mitunter großen Textmengen zur Forschungsfrage führt, ob und wie der Vergleich interaktiv und reaktionsschnell gestalten werden kann. Hinzu kommt der Bedarf nach interaktiven Analysemöglichkeiten der Textvarianten, der sich aus dem Umfang des zu betrachtenden Materials,

aber auch der angestrebten Dynamik bei der Berechnung ableitet. Somit stellt sich als weitere Forschungsfrage, wie eine interaktive Analyse der Textvarianten gestaltet und umgesetzt werden kann?

VI. Die Verschiedenartigkeit der Werke und Projekte macht einen weiteren Punkt deutlich: Trotz vieler Gemeinsamkeiten gibt es zum Teil große Unterschiede zwischen den Editionsvorhaben, sodass ein digitales Werkzeug in dieser Domäne an vielen Stellschrauben eine hohe Generizität aufweisen muss, um in mehr als einem Projekt Anwendung zu finden. Die Spezialisierung und Verallgemeinerung sind dabei im Spannungsfeld. Zum einen verlangen die Herausforderungen eines Projektes mitunter spezialisierte Lösungen. Zum anderen sollte das System aber auch für weitere Editionsvorhaben nutzbar bleiben, zumindest mit einem begrenzten und abschätzbaren Anpassungsbedarf. In einem Artikel von Elena Pierazzo von 2019 [114] wird dieses Problem in Bezug auf digitale Editionen aufgegriffen und mit einer Metapher aus der Modewelt veranschaulicht: So entsprechen viele digitale Entwicklungen eher der *Haute Couture*, das heißt, sie sind einzigartig, innovativ, kreativ, teuer und für die einmalige Nutzung bestimmt. Demgegenüber steht mit der *Prêt-à-Porter* – der Konfektionsware – etwas sofort verfügbares, erschwingliches und für die Mehrfachnutzung bestimmtes. Zwar bringt die *Haute Couture* durch Neuentwicklung und Experimente einen gewissen Fortschritt, doch sind für die allermeisten Editionsvorhaben (speziell ohne umfängliche finanzielle Förderung) nur bestehende Lösungen verfügbar. Es stellt sich also die Frage, wie ein System konkret gestaltet werden kann, um die komplexen Anforderungen der Editionsphilologie zu erfüllen und zeitgleich als *Prêt-à-Porter* breiten Einsatz zu finden.

2.4 Notation

Im Folgenden werden einige für diese Dissertation gewählte Begriffe erläutert sowie die genutzte formale Notation eingeführt.

Ein **Token** T ist eine Zeichenkette und repräsentiert einen kleinen Textbestandteil, wie beispielsweise ein Wort, Satzzeichen, Leerraum oder auch XML-Element (siehe Abschnitt 4.2.1). Eine Menge von Token wird mit $\mathbb{T}$ symbolisiert. Zudem wird ein **leeres Token** ϵ bei der Modellierung genutzt, welches nicht in $\mathbb{T}$ enthalten ist, das heißt $\epsilon \notin \mathbb{T}$. Die syntaktische Gleichheit von zwei Token $T \in \mathbb{T} \cup \{\epsilon\}$ und

$T' \in \mathbb{T} \cup \{\epsilon\}$ kann geprüft werden, es gilt entweder $T = T'$ oder $T \neq T'$. Die **Konkatenation von Token** wird mit dem Operator $+$ (beziehungsweise $\sum$) notiert.

Ein **Textsegment** S, oder kurz **Segment**, ist eine Folge von m Token $T_1, \ldots, T_m$, wobei gilt $S = \sum_{i=1}^{m} T_i$. Der Begriff Segment wird im Kontext der Segmentierung einer Textfassung gebraucht, das heißt, wenn die Fassung in kleinere Einheiten wie Sätze oder Absätze zerlegt wird (siehe Abschnitte 4.2.1 und 5.2). Eine Menge von Segmenten wird als $\mathbb{S}$ symbolisiert.

Ein bei der Kollationierung betrachtetes Werk liegt in n **Textfassungen** beziehungsweise **Fassungen** vor, die als F_i mit $1 \leq i \leq n$ bezeichnet werden. Eine Fassung F_i wird im Rahmen der Dissertation je nach Kontext als Folge von n_i Segmenten mit $F_i = \sum_{j=1}^{n_i} S_j$ oder als Folge von m_i Token mit $F_i = \sum_{j=1}^{m_i} T_j$ betrachtet.

Zudem wird der Begriff **Textpassage** genutzt, wobei eine Textpassage meist mit A oder B bezeichnet wird. Analog zu Textsegmenten handelt es sich dabei um Folgen von Token. Der Begriff wird immer dann gebraucht, wenn es sich explizit nicht um Textsegmente handelt, beispielsweise außerhalb des Kontexts von in Segmente zerlegten Textfassungen, oder wenn in Textsegmenten selbst Tokenfolgen spezifiziert werden.

In der Dissertation, insbesondere in Abschnitt 5.4.3 zur Heuristik zur detaillierten Differenzanalyse, wird der Begriff **Edit-Script** verwendet. Ein Edit-Script $ES_{A \Rightarrow B}$ beschreibt dabei eine Abfolge von Operationen, das heißt Änderungen von Token, um eine Textpassage A in eine andere B zu überführen. Die Anwendung eines Edit-Scripts wird mit dem Operator $\oplus$ notiert. Es gilt: $A \oplus ES_{A \Rightarrow B} = B$.
Die vier im Folgenden benannten Operationen der klassischen Levenshtein-Distanz [85] transformieren jeweils einzelne Token T, $T' \neq T$ oder das leere Token ϵ:

- *:copy* – Kopieren eines Tokens: $T \Rightarrow T$
- *:add* – Ergänzen eines Tokens: $\epsilon \Rightarrow T$
- *:del* – Streichen eines Tokens: $T \Rightarrow \epsilon$
- *:sub* – Substituieren eines Tokens: $T \Rightarrow T'$

Kapitel 3

Verwandte Arbeiten

Im vorherigen Kapitel wurden anhand einiger Editionsprojekte wiederkehrende Problemklassen bei der Kollationierung von Textfassungen identifiziert und daraus sechs wesentliche Anforderungen für digitale Werkzeuge im Bereich der Editionsphilologie abgeleitet. Im ersten Abschnitt dieses Kapitels werden nun zunächst die algorithmischen Grundlagen für einen Textvergleich untersucht. Der Vorstellung der Methoden aus der Bioinformatik zur Bestimmung von Editierdistanzen und Sequenzalignments fällt dabei ein besonderes Gewicht zu, bevor darauf aufbauend zu den digitalen Geisteswissenschaften übergeleitet wird. Im zweiten Unterkapitel wird schließlich die konkrete Realisierung der sechs herausgearbeiteten Anforderungen in bestehender Software untersucht, um effektive Lösungen und Schwächen zu identifizieren. Der letzte Teil ist dabei von besonderem Interesse, zeigt er doch das in dieser Dissertation behandelte Innovationspotenzial auf.

3.1 Algorithmische Grundlagen des Textvergleichs

Die Kollationierung unterschiedlicher Fassungen eines Textes ist, algorithmisch gesehen, im Kern ein Abgleich von Zeichenketten. Dieses Grundproblem in seinen verschiedenen Abwandlungen ist für viele Anwendungen abseits der Editionswissenschaften relevant. Dazu gehören beispielsweise der Versionsvergleich von Artikeln in kollaborativen Enzyklopädien oder der Abgleich von geänderten Gesetzestexten und Verträgen, aber auch das Verfolgen von Quellcodeänderungen, und das Auffinden von Gemeinsamkeiten und Unterscheiden zwischen DNA- oder Proteinsequenzen in der Bioinformatik.

Die Untersuchung biologischer Prozesse war und ist ein wesentlicher Innovationstreiber für Algorithmen zum Textvergleich, auch wenn sich die Ansätze nur bedingt auf die Untersuchung natürlicher Sprachen übertragen lassen (siehe Abschnitt 3.1.2). Trotz der unterschiedlichen Anwendungsdomänen sind die in der Bioinformatik verwendeten Ansätze eine wichtige Grundlage dieser Arbeit und werden im Folgenden zunächst vorgestellt, bevor in einem zweiten Abschnitt die bestehende Adaption dieser Ansätze auf die Editionswissenschaften diskutiert wird.

3.1.1 Von Editierdistanzen und Sequenzalignments

Beim Abgleich von Sequenzen in der Bioinformatik finden sich die beiden, miteinander in Beziehung stehenden Ziele der Bestimmung von Editierdistanzen und von Sequenzalignments. Eine Editierdistanz beschreibt die Anzahl der notwendigen Operationen, um eine Sequenz (zum Beispiel die Basen eines DNA-Strangs) in eine andere zu transformieren, wobei eine solche Sequenz notwendiger Operationen im Rahmen dieser Dissertation als *Edit-Script* bezeichnet wird. Bei einem Sequenzalignment hingegen werden die Elemente der einen Sequenz den Elementen der anderen zugeordnet. Editierdistanzen sind somit Maße für die Unterschiedlichkeit zweier Sequenzen und können wiederum als grundlegende Metrik für die Erstellung eines optimalen Sequenzalignments genutzt werden. Die Repräsentation als Edit-Script oder Sequenzalignment ist dabei, mathematisch gesehen, äquivalent [56, S.217]. Die beiden Probleme werden im Folgenden für die Anwendung auf genau zwei Sequenzen mit konkreten Algorithmen sowie Laufzeiten näher diskutiert, bevor schließlich die Erweiterung der Betrachtung auf mehr als zwei Sequenzen in Form multipler Sequenzalignments folgt.

Editierdistanzen

Editierdistanzen sind Metriken für die Unterschiedlichkeit zweier Zeichenketten. Ihr Wert entspricht der minimalen Anzahl der notwendigen Operationen, um die erste Zeichenkette in die zweite zu überführen. Mitunter werden den Operationen auch verschiedene Kosten zugeordnet, sodass die Editierdistanz deren Summe beschreibt. Die dabei wohl geläufigste und teils als Synonym für Editierdistanz verwendete Metrik ist die Levenshtein-Distanz [85], welche als Operationen das Kopieren, Einfügen, Löschen sowie Ersetzen eines einzelnen Zeichens erlaubt, wobei dem Kopieren Kosten von 0 und den übrigen Operationen Kosten von 1 zugeordnet werden. Das

Maß lässt sich analog für zwei Textpassagen bestimmen, indem die Operationen auf ganze Token, also beispielsweise Wörter, angewendet werden.

Die Levenshtein-Distanz $Lev(A, B)$ zwischen zwei Zeichenketten A und B sowie ein dazugehöriges Edit-Script – eine Abfolge von Operationen, um A nach B zu transformieren – können mit dem als Wagner-Fischer-Algorithmus bekannt gewordenen Ansatz auf Basis dynamischer Programmierung in $O(|A| \cdot |B|)$ bestimmt werden [150]. Eine Verbesserung der Laufzeit auf $O((1+Lev(A, B)) \cdot min(|A|, |B|))$[1] wurde 1985 von Esko Ukkonen vorgestellt [146]. Da die Laufzeit dabei insbesondere von der Levenshtein-Distanz $Lev(A, B)$ selbst abhängt, ist der Ansatz besonders effizient für sehr ähnliche Zeichenketten mit einer entsprechend geringen Levenshtein-Distanz.

Es gibt viele weitere Ansätze zum Bestimmen von Editierdistanzen. Zum Beispiel erreicht ein 2003 von Heikki Hyyrö vorgestellter Algorithmus mittels Parallelität auf Bit-Ebene eine asymptotische Laufzeit von $O(\sigma+\lceil Lev(A, B)/w \rceil \cdot max(|A|, |B|))$, wobei σ die Größe des Zeichenvorrats und w die Wortbreite des verwendeten Computers bezeichnen [70]. Dieser Ansatz ist für Anwendungen mit einem kleinen Zeichenvorrat, wie dem Vergleich von DNA-Strängen oder Sequenzen aus Aminosäuren, sehr vielversprechend, scheidet für Anwendungen mit großem Zeichenvorrat hingegen aus.

Zudem gibt es verschiedene Variationen der Levenshtein-Distanz, welche modifizierte Kosten oder eine abweichende Menge von Operationen erlauben. Ein Beispiel ist die Damerau-Levenshtein-Distanz [34], bei der zusätzlich das Vertauschen benachbarter Elemente als eine mögliche Operation definiert wird. Zwar wird das Problem für den allgemeinen Fall dadurch NP-Vollständig [149], doch unter der Annahme bestimmter Kosten für die Operationen ändert eine entsprechende Modifikation der genannten Algorithmen deren asymptotische Laufzeit nicht [151, 146, 70]. Durch die Beschränkung der möglichen Operationen auf Einfügungen und Löschungen wird aus der Levenshtein-Distanz die Länge der Longest-Common-Subsequence (längsten gemeinsamen Teilsequenz); durch die Beschränkung auf Substitutionen (für Zeichenketten gleicher Länge) wiederum die Hamming-Distanz [71].

[1] In [146] wird die asymptotische Laufzeit mit $O(Lev(A, B) \cdot min(|A|, |B|)$ angegeben, was allerdings den Randfall für identische Zeichenketten und damit $Lev(A, B) = 0$ unbeachtet lässt.

Paarweises Sequenzalignment

Bei einem paarweisen Sequenzalignment werden die Elemente einer Sequenz A den Elementen einer Sequenz B zugeordnet, wobei die Reihenfolge erhalten bleibt, aber Lücken (Gaps) erlaubt sind. Diese Lücken stehen dabei für neu hinzugekommene beziehungsweise für fehlende Teilsequenzen. Einander zugeordnete Elemente können wiederum übereinstimmen (Matches), müssen dies aber nicht, sondern können auch veränderte Teilsequenzen repräsentieren (Mis-Matches). Somit lassen sich analog zur Levenshtein-Distanz die vier Operationen des Einfügens, Löschens, Kopierens und Ersetzens wiederfinden. Unter Zuhilfenahme von ähnlichkeits- beziehungsweise distanzbasierten Bewertungsfunktionen für die entstehenden Matches, Mis-Matches und Gaps wird aus dem Bestimmen eines Sequenzalignments schließlich ein Optimierungsproblem, bei dem das Alignment derart optimiert wird, dass die Bewertungsfunktion einen maximalen beziehungsweise minimalen Wert annimmt. Komplexe Bewertungsfunktionen enthalten dabei Scoring-Matritzen, welche allen möglichen Paaren von Elementen Werte zuordnen.
Ein optimales Sequenzalignment zwischen A und B kann mittels dynamischer Programmierung in $O(|A| \cdot |B|)$ bestimmt werden, wobei dieser Ansatz als Needleman-Wunsch-Algorithmus [101] bekannt wurde und ein sogenanntes *globales* Sequenzalignment bestimmt. Dabei fließt die Zuordnung aller Elemente in die Bewertung ein. Demgegenüber stehen *lokale* Sequenzalignments, bei denen zwei möglichst gute Teilsequenzen (anhand einer ähnlichkeitsbasierten Bewertungsfunktion) aligniert werden und somit nicht alle Elemente im Endergebnis enthalten sind, was für manche biologische Fragestellungen sinnvoll ist. Sie können mittels des Smith-Waterman-Algorithmus [141] bestimmt werden. Lokale Sequenzalignments entsprechen allerdings nicht dem im Rahmen dieser Dissertation untersuchten Anwendungsszenario, bei dem Textvarianten auf voller Länge der Textfassungen eines Werkes untersucht werden sollen, und werden daher im Folgenden nicht weiter betrachtet. Neben den optimalen Berechnungsverfahren gibt es eine Vielzahl von performanten Heuristiken [60].

Multiples Sequenzalignment

Das Bestimmen multipler Sequenzalignments, das heißt der Vergleich von mehr als zwei Zeichenketten beziehungsweise Sequenzen, ist für viele biologische Forschungsfragen relevant. Entsprechend handelt es sich dabei um ein breites und ak-

tives Forschungsthema in der Bioinformatik. In [56, S.335f.] werden beispielsweise drei miteinander in Beziehung stehende Anwendungen beschrieben, nämlich die Repräsentation von Proteinfamilien (oder auch Superfamilien), die Identifikation und Repräsentation von zu einer gemeinsamen Funktion korrelierenden DNA- oder Proteinsequenzen und das Nachvollziehen evolutionärer Prozesse mittels der Generierung phylogenetischer Bäume. Da auch bei der angestrebten Kollationierung mehr als zwei Textfassungen gleichzeitig kollationiert werden sollen, sind die Ansätze zum Bestimmen multipler Sequenzalignments auch für diese Dissertation ein besonders relevantes Forschungsfeld und werden hier kurz wiedergegeben.

Durch die vielfältigen, mithilfe von multiplen Sequenzalignments untersuchten Forschungsfragen werden entsprechend vielfältige Ansätze in der Literatur beschrieben, was auch damit einhergeht, dass die Frage nach der Bewertung eines Alignments und damit des Optimierungsproblems sehr unterschiedlich beantwortet wird[2]. Ein naheliegender und verbreiter Ansatz ist es, die alignierten Sequenzen paarweise mittels einer der bereits genannten Bewertungsfunktionen zu betrachten und diese Einzelergebnisse aufzusummieren, den sogenannten Sum-Of-Pairs-Kosten. Andere Ansätze suchen nach möglichst großer Übereinstimmung (Konsensfunktionen) oder dienen zum direkten Aufbau phylogenetischer Bäume. Die Wahl der Bewertungsfunktion hat wiederum Auswirkungen auf die algorithmische Realisierung und Optimierung, was in der erwähnten Vielfalt der Ansätze zum Bestimmen multipler Sequenzalignments mündet. Zusätzlich besteht analog zu paarweisen Sequenzalignments auch hier die Unterscheidung zwischen globalen und lokalen multiplen Alignments, wobei im Folgenden nur erstere betrachtet werden.

Zur Bestimmung eines optimalen multiplen Sequenzalignments von n Zeichenketten mit einer mittleren Länge von l kann der oben bereits erwähnte Ansatz mittels dynamischer Programmierung auf mehr als zwei Zeichenketten erweitert werden, wie bereits 1985 von Murata et al. auf Basis einer paarweise-summierten Bewertungsfunktion geschehen [96]. Dieser Ansatz hat allerdings eine Laufzeit in $O(l^n)$, auch wenn sich die Anzahl der Berechnungen und der notwendige Speicherplatz für den praktischen Einsatz optimieren lassen [25]. Neueste Ansätze erreichen dabei sogar einen polynomiellen Speicherverbrauch [67]. Dennoch, das dazugehörige

[2] Siehe beispielsweise in [56, S.342f.]: „To date, there is no objective function that has been as well accepted for multiple alignment as (weighted) edit distance or similarity has been for two-string alignment. In fact, [...] some popular methods to find multiple alignments do so without using any explicit objective function. The goodness of those methods is judged by the biological meaning of the alignments that they produce, and so the biological insight of the evaluator is of critical importance.“.

Entscheidungsproblem ist NP-Vollständig [152], sodass eine optimale Berechnung für praxisnahe Eingabegrößen im Allgemeinen ausscheidet.
Um dem Problem der Effizienz zu begegnen, wurde eine Vielzahl heuristischer Ansätze entwickelt, welche wiederum dem Zielkonflikt zwischen der Laufzeit und Güte der berechneten Lösung unterschiedlich begegnen [105]. Neben probabilistischer Ansätze, beispielsweise auf Basis von Hidden Markov Models [42] oder genetischer Algorithmen [28], sind dies insbesondere progressive Alignierungsmethoden. Das grundsätzliche Vorgehen, wie beispielsweise in [144] beschrieben, gliedert sich dabei in zwei Phasen: Zunächst folgt der Aufbau eines *guide trees*, welcher die Ähnlichkeitsbeziehungen zwischen den Sequenzen beschreibt. Darauf aufbauend wird anschließend eine Alignierung der zwei demnach ähnlichsten Sequenzen bestimmt, welche dann schrittweise um die weiteren Sequenzen erweitert wird. Dieser Ansatz wurde bereits 1984 von Hogeweg und Hesper [65] beziehungsweise auf dessen Grundlage 1987 von Feng und Doolittle [45] beschrieben. Um dabei effizient zu sein, können die Ähnlichkeitswerte in der ersten Phase nicht paarweise mittels optimaler Alignierung bestimmt werden. Stattdessen werden beispielsweise in Clustal [27] – einer Reihe verbreiteter Programme für die Bestimmung multipler Sequenzalignments – approximative Verfahren verwendet. Zum anschließenden Aufbau des *guide trees* kommen dann Neighbor-Joining [132] oder verschiedene Clustering-Verfahren zum Einsatz. Auch für die Realisierung der zweiten Phase existieren verschiedene Varianten. So hatten erste Implementierungen zunächst das Problem, dass sich Fehler der bestehenden Alignierung mit jeder hinzukommenden Sequenz verstärken können. Diesem Problem wird mit einer Vielzahl von Abwandlungen der zweiten Phase, beispielsweise mit *iterativen* und *konsistenzbasierten* Methoden [144] sowie der Kombination verschiedener Alignierungsansätze [104], begegnet. Der grundlegende Ansatz der progressiven Alignierung wurde in den Kontext der Kollationierung von Textfassungen übertragen, wie im folgenden Abschnitt ausgeführt wird.

3.1.2 Methodik in den Digitalen Editionswissenschaften

Die Digitalisierung hat seit Langem auch direkten Einfluss auf die Methodik in den Editionswissenschaften, wie sich beispielsweise an der vielfältigen Entwicklung von Algorithmen zur automatischen Kollationierung verschiedener Textfassungen seit den früher 1960er-Jahren erkennen lässt [108]. In ihrer umfangreichen Untersuchung listet Nury in [106] allein 25 verschiedene Werkzeuge zur Kollationierung,

die zwischen 1962 und 2016 entstanden sind. Im Folgenden werden zunächst einige Gemeinsamkeiten und Unterschiede zwischen der Alignierung biologischer Sequenzen und Textfassungen aufgezeigt, bevor näher auf das Vorgehen in den digitalen Geisteswissenschaften eingegangen wird. Dabei wird das Gothenburg-Modell vorgestellt, welches den Prozess der Kollationierung in verschiedene Schritte einteilt. Auf diese Schritte wird wiederum in diesem und dem nächsten Kapitel mehrfach Bezug genommen. Anschließend werden die grundlegenden Vorgehen aufgezeigt, die zur Alignierung von Textfassungen genutzt werden.

Abgrenzung zur Bioinformatik

Zwischen dem Vergleich von als Zeichenketten kodierten biologischen Informationen und in natürlichen Sprachen verfassten Texten gibt es Überschneidungen, aber auch große Unterschiede. So bestehen die in der Bioinformatik untersuchten Sequenzen, genauer Desoxyribonukleinsäure (wobei die englische Abkürzung DNA für Deoxyribonucleic Acid gebräuchlich ist), aus den vier Basen Adenin (A), Guanin (G), Cytosin (C) und Thymin (T). Der Vergleich wird mitunter aber auch auf der Ebene von Aminosäuren unter Zuhilfenahme fest codierter Scoring-Matritzen, wie PAM250 [35] oder BLOSUM62 [62], durchgeführt, wodurch sich der Zeichenvorrat auf 20 beziehungsweise 22 [11] erhöht. Komplette Genome können wiederum mehrere Milliarden Basenpaare aufweisen – beim menschlichen Genom sind es beispielsweise um die drei Milliarden [147]. Allerdings werden Algorithmen für das Sequenzalignment üblicherweise nicht auf vollständigen Genomen ausgeführt. Dafür werden beim multiplen Sequenzalignment oft Hunderte oder einige tausend DNA-Sequenzen miteinander verglichen.
Im Gegensatz dazu ist der zu handhabende Zeichenvorrat in natürlichen Sprachen wesentlich größer. Beim Vergleich auf Wortebene kommen der ganze Wortschatz einer Sprache und damit Zehntausende Entitäten in Betracht. Selbst auf Zeichenebene kann es sich um Dutzende handeln, denn neben den eigentlichen Buchstaben des Alphabets – in Groß- und Kleinschreibung – kommen Diakritika, Satzzeichen sowie beliebige Sonderzeichen hinzu. Auch fremdsprachige Einschübe (in anderen Alphabeten) oder unterschiedliche Kodierungsvarianten der Zeichen können auftreten. Im Gegenzug sind von Menschen verfasste Texte im Vergleich zu vollständigen Genomen relativ kurz und umfassen beispielsweise um die zehn- bis hunderttausend Wörter bei der Betrachtung ganzer Bücher. Die Anzahl der zu vergleichenden Textfassungen variiert hingegen sehr stark und kann je nach Editionsprojekt von sehr

wenigen Exemplaren (*Histoire des deux Indes*), über viele Dutzend (*Keter Shem Ṭov*) bis hin zu Tausenden (Griechische Fassungen des *Neuen Testaments* [111]) reichen. Durch diese quantitativen Unterschiede ist die Adaption der bestehenden Alignierungsverfahren für DNA- und Protein-Sequenzen auf die Kollationierung von Textfassungen nur eingeschränkt möglich. Gerade die unterschiedliche beziehungsweise variable Größe des Zeichenvorrats macht fest-codierte Scoring-Funktionen und damit einen wesentlichen Faktor für die Effizienz vieler bioinformatischer Algorithmen für natürliche Sprachen unmöglich.
In [68] werden allerdings auch qualitative Gemeinsamkeiten beim Abgleich der Evolution von Genomen und von Traditionslinien von Manuskripten identifiziert. Demnach finden sich in beiden Anwendungsgebieten ähnliche Phänomene, wie beispielsweise die Rekombination, die konvergente Evolution oder die Änderung der Reihenfolge von Teilsequenzen. Sowohl in der Evolutionsforschung als auch der Stemmatologie haben sich Baumstrukturen zur Darstellung der Abstammungsverhältnisse etabliert. In der Editionsphilologie wird diese als *Stemma Codicum* bezeichnete Darstellungsform bereits seit zwei Jahrhunderten verwendet [64]. Während computergestützte Verfahren für die Geisteswissenschaften allerdings zunächst noch in den Kinderschuhen steckten, hatten diese Methoden die biologische Forschung bereits deutlich verändert, wie Roos und Zou in einem Artikel von 2011 [130] anmerken[3]. So wurde in der stemmatologischen Forschung zunächst auch mit bestehenden Verfahren der Bioinformatik experimentiert, siehe zum Beispiel [68], bevor gezielt digitale Werkzeuge für die Geisteswissenschaften, wie Stemmaweb [9], entwickelt wurden.

Gothenburg Modell

Das Gothenburg Modell geht auf eine gemeinsame Initiative der Entwickler:innen von CollateX und Juxta zurück, die 2009 im Rahmen eines Workshops in der namensgebenden Stadt zusammen kamen [38, 92]. Es strukturiert den Prozess der automatischen Kollationierung mit dem Ziel, eine übergreifende theoretische Grundlage zu schaffen, welche die Entwicklung entsprechender Werkzeuge oder einzelner Komponenten sowie deren Zusammenarbeit vereinfacht. Das vollständige Modell beinhaltet nach [38] fünf verschiedene Aufgaben, nämlich die Tokenisierung, Nor-

[3] Aus dem Englischen: „The adoption of computational methods in textual criticism, and in humanities at large, is still in its infancy. [...] In contrast, the methodology of the biological sciences has been utterly transformed by mathematical and computational methods;“ [130].

malisierung, Alignierung, Analyse und Visualisierung. Es gibt allerdings abweichende Beschreibungen, welche beispielsweise die Tokenisierung und Normalisierung als einen Schritt zusammenfassen [37] oder den Analyseschritt nur implizieren [72]. In einer Modifikation von Nassourou 2013 wird das Modell mit der Interpretationsschicht sogar um einen Schritt erweitert [98]. Doch allen Beschreibungen gemein ist der beinhaltete Schritt der algorithmischen Alignierung der Textfassungen.

Umsetzung der Alignierung

Wie bereits aus der Betrachtung (multipler) Sequenzalignments deutlich wurde, ist die Bestimmung optimaler Alignierungen algorithmisch komplex und führt zu entsprechend hohen Laufzeiten. Dennoch gibt es unter den Kollationierungswerkzeugen einige Ansätze, welche optimale Verfahren umsetzen. Dazu gehört der in TRAViz [75] integrierte Sentence Alignment Algorithmus[4], welcher beliebig viele Textfassungen nach dem Brute-Force-Prinzip vergleicht, was ihn entsprechend nur für kleine Textmengen praktikabel macht.
Andere optimal arbeitende Verfahren beschränken den Vergleich auf genau zwei Textfassungen, beispielsweise das Kollationierungstool Variance-Viewer [13] auf Basis des Algorithmus von Myers [97], oder arbeiten nach dem klassischen Leitquellenprinzip, bei dem der optimale Vergleich der Textfassungen immer in Bezug auf einen zuvor gewählten Basistext stattfindet (mehr dazu in Abschnitt 3.2.1).
Aus Effizienzgründen wurden auch im Rahmen der Kollationierung von Textfassungen verschiedene heuristische Ansätze entwickelt. Dazu gehört der Sliding-Window-Ansatz, bei dem die Textfassungen nur innerhalb eines Fensters bestimmter Größe miteinander verglichen werden und dieses nach und nach über die Textfassungen bewegt wird. Das Verfahren orientiert sich dabei an der manuell durchgeführten Kollationierung und wurde in einer Vielzahl früher Programme ab 1962 implementiert, wie Schmidt in [136] ausführt. Aufgrund der wesentlichen Schwäche des Verfahrens, dass Gemeinsamkeiten zwischen den Textfassungen außerhalb des Fensters

[4] Der Algorithmus wird in [74] genauer beschrieben: Er hat die Aufgabe, einen Variantengraphen für eine Menge von Textfassungen zu bestimmen. Dazu wird für jede Textfassung ein Kettengraph generiert, wobei die Token – in der Regel die Wörter – der Textfassung die Knoten bilden und diese für zwei im Text aufeinander folgende Token durch eine gerichtete Kante verbunden werden. Nun werden sukzessive die Knoten für zwei gleiche Token aus unterschiedlichen Textfassungen verschmolzen, insofern dadurch kein Kreis entsteht. Der Algorithmus probiert ausgehend vom initialen Zustand alle gültigen Abfolgen von Verschmelzungsoperationen aus und wählt schließlich den Variantengraphen als Ergebnis aus, bei dem die meisten Knoten zusammengefügt werden konnten.

nicht erkannt werden, sowie schwindender Limitierungen durch die Weiterentwicklung der Rechentechnik, wird es in aktueller Software nicht mehr verwendet.
Der aus dem Kontext des multiplen Sequenzalignments in der Biologie stammende heuristische Ansatz der progressiven Alignierung [65, 45] wurde auch für die Kollationierung von Textfassungen adaptiert. Eine erste Beschreibung dessen findet sich 2004 bei Spencer und Howe [143], wobei allerdings die Editierdistanz zwischen allen Paaren von Textfassungen berechnet wird, um den *guide tree* auszubauen, was wiederum dem heuristischen Grundgedanken widerspricht. Im Kollationierungswerkzeug CollateX [37] wird ebenfalls der progressive Ansatz verfolgt, wobei die Textfassungen allerdings ohne vorherige Ähnlichkeitsanalyse, das heißt, in gegebener Reihenfolge zum bisherigen Alignierungsergebnis hinzugefügt werden.

3.2 Tools in Abgleich mit den Anforderungen

Im Folgenden werden die sechs wesentlichen, in Abschnitt 2.3 zusammengetragenen Anforderungen und Bedarfe an digitale Werkzeuge für die Editionsphilologie mit bestehenden Werkzeugen und Visualisierungen dieser Domäne abgeglichen, um effektive Lösungen, aber auch bestehende Schwächen aufzuzeigen. Die sechs Anforderungen werden dazu in weitere Unterpunkte aufgeteilt und nacheinander diskutiert. Dabei handelt es sich um:

I. Vergleich von mehreren Textfassungen
   (a) Handhabung von mehr als zwei Textfassungen
   (b) Realisiertes Vergleichsprinzip
   (c) Umgang mit vielen (Hunderten) Textfassungen

II. Vergleich von umfangreichen Textfassungen
   (a) Verarbeitbare Textmenge
   (b) Ein- oder mehrstufige Kollationierung
   (c) Skalierende Visualisierungsformen

III. Umgang mit komplexen Textvarianten
   (a) Normalisierungsoptionen

(b) Umgang mit Transpositionen

(c) Typografie, Semantik und Mehrsprachigkeit

IV. Nutzerfreundlichkeit des Systems

(a) Installation

(b) Nutzerführung

(c) Interoperabilität

V. Interaktivität des Systems

(a) Interaktive Bestimmung von Textvarianten

(b) Interaktive Analyse der Textvarianten

VI. Generizität des Systems

(a) Lösungen für die vier Zielsetzungen der Kollationierung laut 2.2.3

(b) Unterstützte Sprachen

3.2.1 Vergleich von mehreren Textfassungen

Handhabung von mehr als zwei Textfassungen (I-a)

Die Kollationierung von mehr als zwei Textfassungen ist für viele Editionsprojekte ein wesentlicher Bedarf, wird aber von einigen digitalen Kollationierungswerkzeugen nicht unterstützt. Dazu gehören beispielsweise TEI Comparator [31], Variance-Viewer [13] oder Colato [80] sowie das klassische Konsolenwerkzeug diff [69], mit welchen jeweils nur zwei Textfassungen direkt miteinander verglichen werden können. Informationen zu den Gemeinsamkeiten und Unterschieden zwischen allen Fassungen müssen bei Verwendung dieser Werkzeuge mit Zusatzaufwand (manuell oder mithilfe einer darauf aufbauenden Softwarelösung) zusammengetragen werden.
Auch mehrere verbreitete Darstellungsformen sind auf den Vergleich von genau zwei Textfassungen ausgelegt. Dazu gehört beispielsweise die in Juxta [**Wheeler2013**] implementierte *Side-by-Side*-Visualisierung, welche die beiden Texte als Synopse anordnet und in Abbildung 3.1 zu sehen ist. Hierbei werden die beiden Fassungen im Volltext einander gegenübergestellt und die Textvarianten direkt in beiden Texten

farblich hervorgehoben. Die Position einer Textvariante in beiden Texten wird dabei durch eine zusätzliche, beide Seiten verbindende Linie veranschaulicht. Diese deutet mittels der Farbintensität zudem das Ausmaß der Variation an.

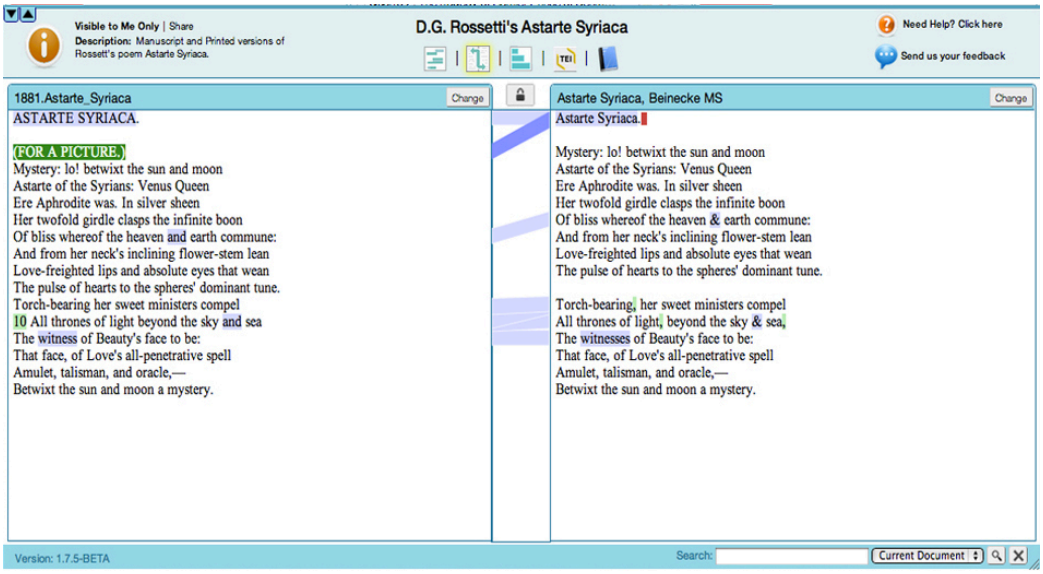


Abbildung 3.1: Die *Side-by-Side*-Visualisierung von Juxta (Web Services) zum Vergleich zweier Textfassungen. Der Screenshot wurde auszugsweise von [103] übernommen.

Diese Art der direkten, synoptischen Gegenüberstellung von Textfassungen, entweder von Volltext als *Side-by-Side*-Visualisierung oder in eher abstrahierter Form als *Aligned Barcode*, wird häufig zur Darstellung alignierter Texte genutzt, wie Yousef und Jänicke in einem Übersichtsartikel von 2021 zur Visualisierung alignierter Texte aufzeigen [156]. Durch Aneinanderreihung lassen sich beide Visualisierungsformen auch indirekt auf mehr als zwei Textfassungen anwenden, was insbesondere sinnvoll ist, wenn die Textfassungen nach einer vorgegebenen Reihenfolge paarweise miteinander kollationiert wurden und der Fokus entsprechend auf den Textvarianten von direkt benachbarten Textfassungen liegt. Eine dritte verbreitete Form der Visualisierung für genau zwei Textfassungen sind zweidimensionale Matrizen, bei Yousef und Jänicke als *Grid-based Heat Maps* bezeichnet [156]. Ein Paar aus zwei verglichenen Textstellen entspricht dabei einer Zelle in der Matrix, die entsprechend der Ähnlichkeit eingefärbt wird.

Realisiertes Vergleichsprinzip (I-b)

Eine Mehrheit der untersuchten Ansätze erlaubt die direkte Handhabung von mehr als zwei Textfassungen und dies sowohl in Bezug auf die Berechnung als auch die Darstellung der Kollationierung. Das wirft wiederum die Frage nach dem verwendeten Vergleichsprinzip auf. Bei vielen Ansätzen kommt das klassische Leitquellenprinzip aus dem Printbereich zum Einsatz, bei dem die Textfassungen ausschließlich mit einem ausgewählten Basistext verglichen werden. Nach diesem Prinzip arbeitet das bereits erwähnte Juxta, bei dem mehrere Fassungen als die *Vergleichsmenge* festgelegt werden können, woraus der Basistext vor der eigentlichen Kollationierung gewählt wird [102]. Auch das Werkzeug eComparatio [16] sowie TUSTEP [109] mit seinen *#VERGLEICHE*- und *#VAUFBEREITE*-Funktionen (siehe Nutzerhandbuch [133]) setzen die Wahl eines Basistextes voraus.
Einige digitale Werkzeuge übertragen dabei die klassische Darstellungsform des Variantenapparats aus dem Buchdruck, der die Unterschiede der Textfassungen ausschließlich bezüglich des Basistexts möglichst platzsparend aufführt. Andere nutzen den verfügbaren Platz der digitalen Medien und wiederholen die sich unterscheidenden Textstellen für alle Fassungen in Gänze.
Die in Juxta integrierte und in Abbildung 3.2 zu sehende *Heatmap* ist eine ebenfalls nach dem Leitquellenprinzip arbeitende Visualisierung. In ihr werden in der Volltextdarstellung der gewählten Fassung die Stellen mit Textvarianten farblich hervorgehoben. Über die Farbintensität wird dabei veranschaulicht, wie viele der anderen Fassungen vom Basistext abweichen. Es lassen sich kleine Apparate via Popup-Fenster für die einzelnen Textstellen hinzu schalten.
Ein anderes Vorgehen als das Leitquellenprinzip wird in CollateX realisiert, welches eine progressive Alignierung zwischen den Textfassungen bestimmt und Nutzer:innen dazu zwischen drei Algorithmen (Dekker, Needleman-Wunsch, MEDITE) wählen lässt [36]. Dieses Vorgehen bei der Alignierung hat den wesentlichen Vorteil, dass die Textvarianten zwischen allen Textfassungen bestimmt werden. Allerdings kann das Ergebnis der progressiven Alignierung stark von der gewählten Reihenfolge der Textfassungen abhängen, sodass auch hier eine gewisse Priorisierung der Textfassungen mit einfließt. Eine Abschwächung dieses Problems, beispielsweise durch eine vorgeschaltete Analyse, um eine möglichst gute Reihenfolge zu wählen, ist für CollateX laut Webseite angedacht, wurde bisher allerdings noch nicht umgesetzt [38].
Ein tatsächlich gleichrangiges Vergleichsprinzip bei der Kollationierung der Textfas-

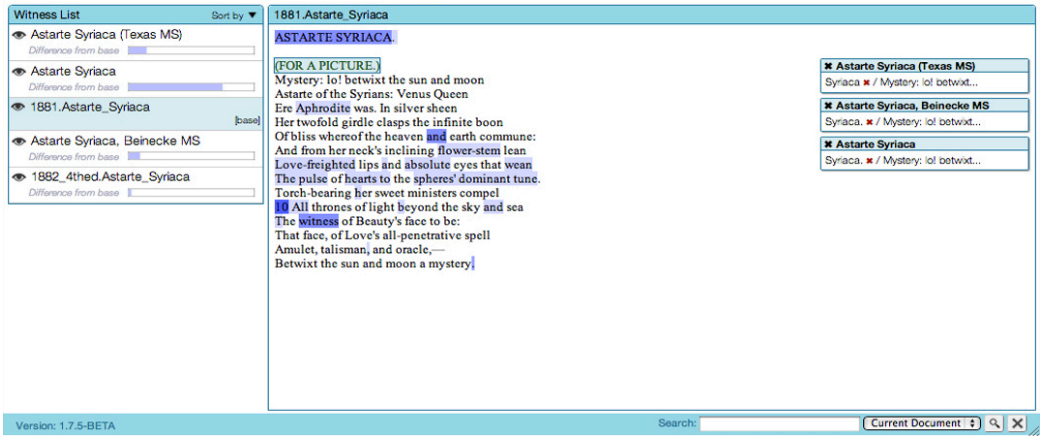


Abbildung 3.2: Die *Heatmap*-Visualisierung von Juxta (Web Services), mit der die Textvarianten bezüglich eines Basistextes farblich in dessen Volltextdarstellung angezeigt werden. Der Screenshot wurde auszugsweise von [103] übernommen.

sungen wurde mit Ausnahme von TRAViz [75] bei keinem der untersuchten Werkzeuge realisiert. Der in TRAViz integrierte Algorithmus arbeitet allerdings nach der Brute-Force-Methode und ist daher nur für sehr kleine Textsegmente ausgelegt. Zudem ist anzumerken, dass auch hier eine unterschiedliche Eingabereihenfolge der Textfassungen zu unterschiedliche Ergebnissen führen kann; nämlich genau dann, wenn es verschiedene optimale Alignierungen, das heißt Abfolgen mit gleich vielen Verschmelzvorgängen, gibt, wie in Abbildung 3.3 an einem Trivialbeispiel veranschaulicht.

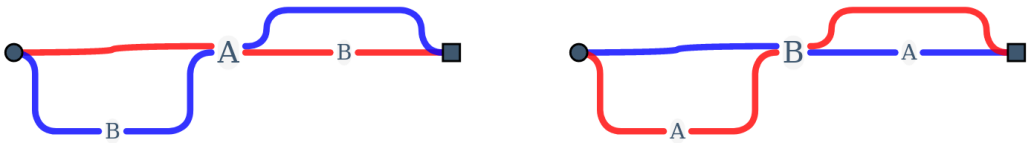


Abbildung 3.3: Zwei mittels TRAViz generierte und visualisierte Variantengraphen für die Textfassungen „A B“(rot) und „B A“(blau). Je nach Eingabereihenfolge der Textfassungen unterscheiden sich die Ergebnisse.

Dem gegenüber stehen vielfältige Ansätze zur gleichrangigen Visualisierung der Textvarianten mehrerer Textfassungen. Eine weit verbreitete Form der Modellierung sind die 2009 von Schmidt und Colomb im Kontext der Kollationierung von

Textfassungen eingeführten Variantengraphen [137]. Dabei handelt es sich um gerichtete azyklische Graphen mit einem Start- sowie Endknoten und Kanten, an denen sich Bezeichner für die Textfassungen befinden. Je nach Realisierung sind die Kanten oder Knoten mit Textpassagen versehen. Folgt man nun einem Pfad anhand des Bezeichners genau einer Textfassung und konkateniert die dabei passierten Textpassagen, so entsteht der Inhalt der Textfassung. Variantengraphen können direkt für die Darstellung der Kollationierungsergebnisse genutzt werden. So nutzt CollateX Variantengraphen nicht nur intern als Datenstruktur für die Algorithmen zur progressiven Alignierung, sondern ermöglicht auch deren Ausgabe in zwei maschinenlesbaren Formaten. Auf der Webseite der Demoversion werden sie zudem als Ergebnis einer Kollationierung direkt visualisiert, wie anhand eines Beispiels in Abbildung 3.4 illustriert wird.

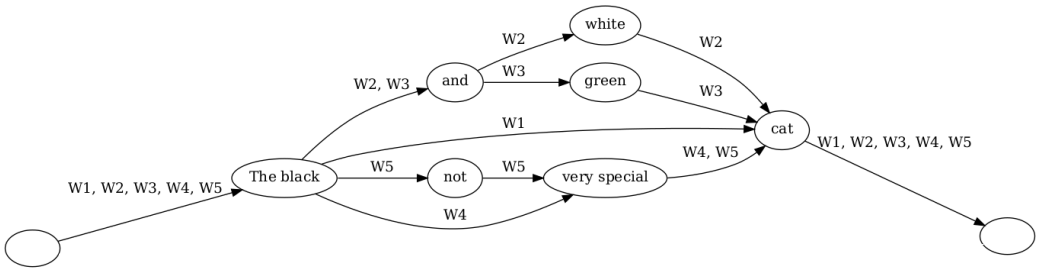


Abbildung 3.4: Ein mittels der Demoversion von CollateX (`https://collatex.net/demo`) generierter und visualisierter Variantengraph.

Mit dem bereits erwähnten TRAViz existiert eine erweiterte Realisierung von Variantengraphen zur Visualisierung von Kollationierungsergebnissen. Die Javascript-Bibliothek nutzt dabei parallel verlaufende Kanten, deren Farben für die Textfassungen stehen. Die angepasste Größe der Worte in den Knoten gibt einen weiteren optischen Hinweis zur Übereinstimmung der Textfassungen.
Neben den Variantengraphen gibt es weitere verbreitete Visualisierungsformen, die sich für mehr als zwei Textfassungen gleichzeitig eignen. Zum einen sind dies tabellenförmige Darstellungen, wie die Alignierungstabelle von CollateX oder die *Interlineare Darstellung* von eComparatio [17, 16]. In beiden Realisierungen gibt es jeweils eine Zeile pro Textfassung. Die einzelnen Wörter beziehungsweise kurze Textsegmente bilden die Zellen. Wurden diese zwischen verschiedenen Textfassungen einander zugeordnet, werden sie in der selben Spalte angeordnet. Abbildung 3.5

zeigt die tabellarische Darstellung eines kurzen Beispiels in beiden Werkzeugen.

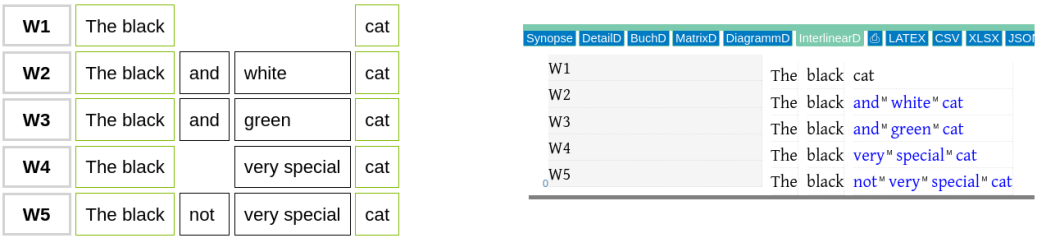


Abbildung 3.5: Tabellarische Darstellung der Vergleichsergebnisse von CollateX (links) und eComparatio (rechts). Während CollateX die Textvarianten zwischen allen Fassungen betrachtet, werden sie in eComparatio nur bezüglich eines gewählten Basistexts (W1) berechnet und dargestellt.

Eine weitere Visualisierungsmöglichkeit sind die in [156] als Sequence-aligned Heat Maps bezeichneten abstrahierten Formen der Alignierungstabellen. In den Tabellenzellen werden dabei geometrische Formen wie Rechtecke statt des Volltexts genutzt. Mittels Farbgestaltung kann dabei das Ausmaß der Variation veranschaulicht werden. Beispiele hierfür sind das Werkzeug TexTile [10], welches die Levenshtein-Distanz einer Textvariante mittels einer sechsstufigen Farbskala codiert, oder auch eine 2009 von Fry erstellte Visualisierung der Unterschiede zwischen sieben Textfassungen von Charles Darwins *On the Origin of Species* [49].

Umgang mit vielen (Hunderten) Textfassungen (I-c)

Es bleibt schließlich die Frage bestehen, wie mit sehr vielen Textfassungen gleichzeitig umgegangen wird.
Einige Ansätze haben bei einer steigenden Anzahl von Textfassungen starke Probleme mit der Effizienz. Dazu gehört beispielsweise der in TRAViz integrierte Alignierungsalgorithmus, der nach dem Brute-Force-Ansatz verfährt. Die untersuchten Ansätze, die nach dem Leitquellenprinzip oder mittels progressiver Alignierung arbeiten, skalieren besser: Kommt eine Textfassung hinzu, muss diese nur mit dem Basistext beziehungsweise mit der bestehenden Alignierung verglichen werden. Dennoch erlaubt Juxta beispielsweise nur den Vergleich von maximal 15 Textfassungen gleichzeitig [103].
Der Umgang mit vielen Textfassungen stellt für CollateX prinzipiell kein Problem dar. Sowohl die Berechnung mittels der drei integrierten Algorithmen zur progres-

siven Alignierung als auch die Darstellung als Variantengraph skalieren problemlos, wenn sie auf 5, 10, 20 oder gar 100 Textfassungen gleichzeitig angewendet werden[5]. Bereits Collate [128], der Vorläufer von CollateX, wurde anhand einer Edition mit 44 Textfassungen entwickelt und kann theoretisch beliebig viele Textfassungen verarbeiten[6].

Neben der Darstellung von Textvarianten werden Graphen auch zur Visualisierung auf Fassungsebene genutzt. Ein Beispiel hierfür sind die Stemmagraphen von Stemmaweb [9], welche die Traditionslinien der Überlieferung eines Werkes nachzeichnen. Hierbei fungieren die zuvor manuell oder mittels Kollationierung bestimmten, stemmatologischen Beziehungen zwischen den überlieferten (und nicht überlieferten) Textfassungen als Kanten des Graphen. Mit jeder Textfassung kann dabei allerdings mehr als ein neuer Knoten hinzukommen, da auch mögliche Zwischenschritte als Knoten repräsentiert werden. Abbildung 3.6 zeigt ein Beispiel.

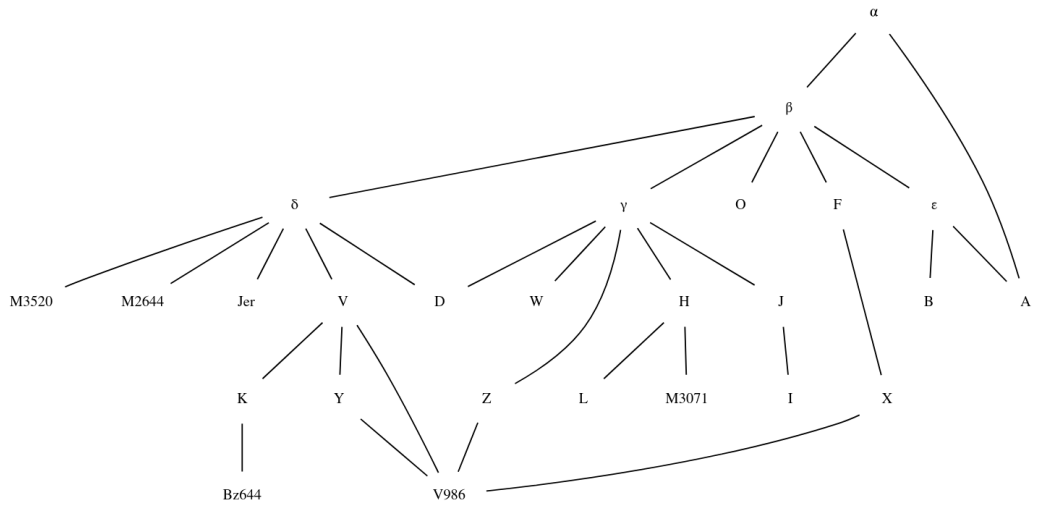


Abbildung 3.6: Ein Stemmagraph für das Werk *Chronicle of Matthew*, übernommen von `https://stemmaweb.net/stemmaweb/`.

Aber nicht alle Darstellungsformen sind für so viele Textfassungen ausgelegt. So wächst die Höhe von Alignierungstabellen, wie jener in CollateX, zwar nur line-

[5] Die Aussage bezieht sich rein auf die Anzahl der Textfassungen und gilt unter der Annahme, dass diese nicht sehr lang oder variantenreich sind.

[6] Nach [128]: „There is no theoretical limit to the number of mss and the length of text COLLATE could process“.

ar zur Anzahl der Textfassungen, doch ab einer gewissen Anzahl überschreiten sie die Bildschirmhöhe und werden unhandlich. Die lineare Größenzunahme gilt auch für die meisten anderen Visualisierungen. Ausnahmen bilden die bereits genannten Variantengraphen und die Darstellung genau einer Textfassung, wie in Juxtas *Heatmap*, bei der die anderen Textfassungen nach dem Leitquellenprinzip nur als farblich codierte Textvarianten auftreten.

3.2.2 Vergleich von umfangreichen Textfassungen

Verarbeitbare Textmenge (II-a)

Die zu vergleichenden Textfassungen können sehr umfangreich sein, wie am Beispiel der *Histoire des deux Indes* deutlich wird. Die verarbeitbare Textmenge bei der Berechnung und Darstellung einer Kollationierung ist also ein weiteres wichtiges Kriterium für entsprechende Werkzeuge.
Einige Werkzeuge sind nicht auf größere Textmengen ausgelegt. So skaliert die Berechnung der Alignierung mittels TRAViz aufgrund des intern verwendeten Brute-Force-Ansatzes für länger werdende Textsegmente sehr schlecht[7].
Für CollateX wird empfohlen, es bis zur Länge eines Kapitels zu verwenden, da die Berechnungsgeschwindigkeit bei größeren Textmengen rapide bis hin zur Nichtnutzbarkeit abnimmt[8]. Ein eigener Test bestätigt dies für alle drei der in CollateX verfügbaren Algorithmen[9].
Das Werkzeug eComparatio ermöglicht die algorithmische Verarbeitung ganzer Bücher. So resultierte ein Test mit der *Histoire des deux Indes* in Berechnungszeiten von um die 20 Minuten (siehe Anhang A).

[7] Eine Stichprobe mit zwei normalisierten Textfassungen der *Histoire des deux Indes* ergabt für einen Auszug mit 500 Wörtern eine Berechnungszeit von rund vier Sekunden, mit 1.000 Wörtern von rund 50 Sekunden, mit 2.000 Wörtern von rund 685 Sekunden (über elf Minuten) und für 4.000 Wörter schließlich rund 14.050 Sekunden (knapp 4 Stunden). Siehe Anhang A.

[8] Aus dem Englischen: „CollateX's performance as to speed may leave something to wish for. Currently, sentence- to paragraph-sized collations are executed in less than a second, virtually independent of the amount of witnesses. Collation of chapter-sized texts is feasible (seconds), but at larger sizes ('book length'), the speed degrades rapidly to unfeasible." [36, S.461].

[9] Wie in Anhang A ausgeführt, erreichen die drei in CollateX verfügbaren Algorithmen für zwei Textfassungen mit einer Länge von 4.000 Wörtern noch Laufzeiten im Bereich von vier bis 30 Sekunden. Bei 8.000 beziehungsweise 32.000 Wörtern sind die MEDITE- beziehungsweise Needleman-Wunsch-Implementierungen nicht mehr anwendbar und stürzen ab, während der Dekker-Algorithmus bei 32.000 Wörtern bereits rund acht Stunden rechnet.

Die verarbeitbare Textmenge in Juxta kann nicht mehr direkt getestet werden, da sich das Werkzeug nicht mehr installieren lässt. Das umfangreichste mitgelieferte Beispiel war der Gedichtsband *Leaves of Grass* von Walt Whitman's mit bis zu 15.000 Wörtern[10] in drei Fassungen [102]. Andere Beispiele waren das erste Kapitel von *Frankenstein* in zwei Fassungen mit gut 8.500 Wörtern und die *Studien zur Geschichte der Renaissance* von Walter Horatio Pater, ebenfalls in zwei Fassungen mit gut 7.500 Wörtern.

Ein- oder mehrstufige Kollationierung (II-b)

Um den Performance-Problemen in der Praxis zu begegnen, wird für Werkzeuge wie Juxta[11] oder CollateX[12] eine Aufteilung der Texte zur Kollationierung und die anschließende Zusammenstellung der separat bestimmten Teilergebnisse empfohlen.

Mit dem hier implizierten Ansatz einer mehrstufigen Kollationierung kann neben dem Performance-Problem noch einer weiteren, grundlegenden Herausforderung beim Edieren sehr umfangreicher Werke begegnet werden, nämlich der notwendigen Wahl der Vergleichsebene: Findet der Vergleich auf Wortebene statt, kann er zu einem wenig hilfreichen, weil überladenen Variantenapparat führen. In Juxtas Beispiel zum ersten Kapitel von *Frankenstein*[13] gibt es starke Überarbeitungen am Ende des Kapitels, sodass fast der gesamte Abschnitt als Variante hervorgehoben wird. Unterbrochen werden diese Hervorhebungen recht willkürlich an Stoppwörtern, wie „a“, „the“, „of“ und „to“. Der daraus resultierenden Apparat ist entsprechend unübersichtlich und wenig hilfreich. Andererseits würden die für eine Textanalyse wichtigen Details auf Wortebene fehlen, wenn der Textvergleich nur auf höheren Vergleichsebenen – wie Zeilen, Sätzen oder Absätzen – durchgeführt wird.

[10] Dies umfasst allerdings auch die mittels <front>-Tag gekennzeichnete Titelei, wobei unklar ist, ob diese ebenfalls Teil der Kollationierung war. Sonst haben die Textfassungen einen Umfang von rund 7.000 Wörtern.

[11] So steht im Nutzerhandbuch von Juxta unter dem Punkt „Preparing Witnesses“ Folgendes: „However, when dealing with large files, or XML-encoded documents with extensive markup, it is possible to create multiple witnesses from one source file.“ [103].

[12] So heißt es in [36, S. 461] bezogen auf die Laufzeit: „[…] chunking or breaking larger bodies of texts into smaller parts is an effective solution.“.

[13] Das Beispiel kann unter `https://web.archive.org/web/20200803144034/http://juxtacommons.org/take_tour` (besucht am 27.01.2024) im Abschnitt „Browse the Gallery“ eingesehen werden; die originale Seite von Juxta ist hingegen nicht mehr erreichbar.

Diesem Problem kann mit einer mehrstufigen Kollationierung begegnet werden, bei der zum Beispiel zunächst größere Segmente aligniert werden und dann in einer zweiten Phase ein detaillierter Vergleich der Wörter beziehungsweise Token der in der ersten Stufe alignierten Segmente stattfindet.
Trotz dieser beiden Vorteile gibt es unter den untersuchten Werkzeugen nur wenige, die den Ansatz einer mehrstufige Kollationierung implementieren und so Nutzer:innen direkt zur Verfügung stellen. Eine Ausnahme bildet das Kollationierungswerkzeug LAKomp [52, 84, 4], welches zunächst eine Alignierung auf (Teil-)Satzebene vornimmt und anschließend die Textvarianten auf Wortebene berechnet.

Skalierende Visualisierungsformen (II-c)

Zusätzlich zu den Laufzeitproblemen der eigentlichen Berechnung wird die Visualisierung mit vielen der oben genannten Werkzeuge bei großen Textmengen schnell unhandlich, insbesondere wenn rein auf Wortebene kollationiert wird. So skalieren Visualisierungen, welche die Textfassungen im Volltext darstellen, naturgemäß sehr schlecht, da mit zunehmender Textmenge auch der Platzverbrauch steigt und die Visualisierung entsprechend zunehmend unübersichtlich wird. Dazu zählen synoptische Volltext-Darstellungen oder Juxtas Heatmap, welche den Basistext im Volltext anzeigt. Aber auch Alignierungstabellen und Variantengraphen eignen sich aus dem selben Grund nur für kurze Texte, wodurch sich die Analyse umfangreicher Werke mit Werkzeugen wie CollateX oder TRAViz schwierig gestaltet.
Die genannten Visualisierungen zeigen (Teile der) Texte direkt an und folgen damit einem Close-Reading-Ansatz. Im Gegensatz dazu gibt es (meist ergänzende) Formen der Distant-Reading-Visualisierung, die stärker von den Textfassungen und ihren Varianten abstrahieren und sich daher für allgemeinere oder strukturelle Übersichten und zur Erforschung globaler Merkmale eignen. Dazu gehören die bereits genannten Formen der Aligned Barcodes und Sequence-aligned Heat Maps nach [156]. Ein anderes Beispiel hierfür ist das *Collation Histogram* von Juxta, welches den Grad an Variation als Balkendiagramm über die Länge des gewählten Basistextes darstellt. Es erleichtert zudem die Navigation im Basistext, indem durch Klick auf einen Balken zur entsprechenden Stelle im Volltext gesprungen werden kann. Dieses Prinzip findet sich ebenso in der Navigations- und Übersichtsleiste CATview (siehe Abschnitt 6.2), welche für das in den folgenden Kapiteln diskutierte Werkzeug LERA entwickelt, aber beispielsweise auch in LAKomp integriert wurde. Hierbei kommt auch der in LAKomp verfolgte Ansatz der mehrstufigen Kollationierung

zum Tragen, indem in einer ersten Stufe zunächst (Teil-)Sätze in synoptischer Gegenüberstellung sowie übersichtlich via CATview angezeigt werden. Nutzer:innen können darauf aufbauend in die zweite Stufe auf Wortebene springen, in welcher die Textvarianten in Form einer Alignierungstabelle (Partitursynopse) veranschaulicht werden. Auch ein in [76] vorgestelltes Konzept-Werkzeug mit dem Namen iteal[14] verfolgt den mehrstufigen Ansatz, in dem es zwei Ebenen von alignierten Barcodes sowie die Detailansicht mittels eines Variantengraphen zur Visualisierung der Textvarianten kombiniert. Abbildung 3.7 zeigt einen Screenshot von iteal zur Illustration.

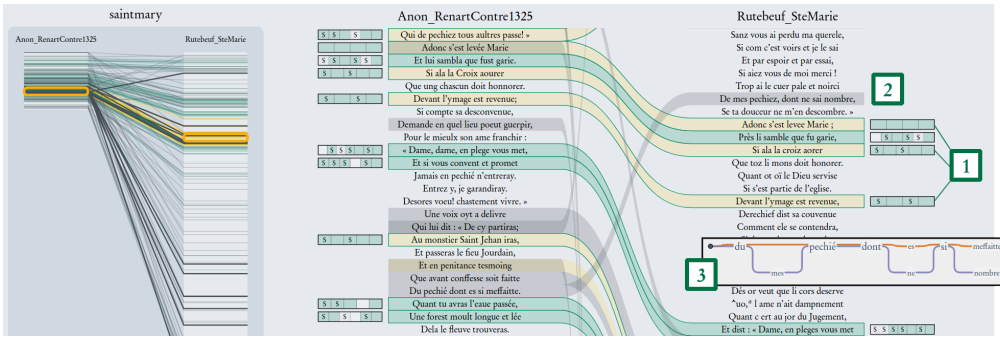


Abbildung 3.7: Screenshot des Werkzeugs iteal, welches drei Distant- und Close-Reading-Visualisierungen hierarchisch miteinander kombiniert, um Textvarianten auf verschiedenen Vergleichsebenen darzustellen. Der Screenshot wurde von [76] übernommen.

Prinzipiell müssen bei zunehmender Textmenge aufgrund der eingeschränkten Bildschirmgröße auch Distant-Reading-Ansätze die Darstellungsgröße der Elemente reduzieren, wenn alle Elemente gleichzeitig dargestellt werden sollen. Allerdings lässt sich dieser Umstand durch eine interaktive Gestaltung, beispielsweise einer geeigneten Zoomfunktion, umgehen, wie in Abschnitt 3.2.5 vertiefend diskutiert wird. Zum Beispiel nutzt TexTile solche Interaktionsmöglichkeiten, um Distant- und Close-Reading in einer einziger Visualisierung miteinander zu vereinen.

[14] Die Ankündigungsseite des Werkzeugs findet sich unter: `https://iteal.vizcovery.org/` (besucht am 17.11.2023).

3.2.3 Umgang mit komplexen Textvarianten

Normalisierungsoptionen (III-a)

Je nach Editionsprojekt und Forschungsfrage ist nicht jede Textvariante von Interesse. So können beispielsweise Unterschiede in der Interpunktion oder Orthographie sprachhistorisch relevant sein, ihre Hervorhebung aber stören, wenn der Fokus auf den inhaltlichen Unterschieden liegt. Die meisten digitalen Werkzeuge bieten daher Funktionalitäten an, um die Textfassungen für die Kollationierung zu normalisieren.
Die *#VERGLEICHE*-Funktion von TUSTEP ermöglicht mittels des Setzens entsprechender Parameter, die Groß- und Kleinschreibung von Buchstaben zu ignorieren und eigene Ersetzungsmuster zu definieren [133].
Juxta bietet ebenfalls vordefinierte Filter für die Normalisierung von Interpunktionen, der Groß- und Kleinschreibung, der Silbentrennung sowie für Leerzeichen an, die über die Nutzeroberfläche ausgewählt werden können [129].
Auch CollateX ermöglicht die Groß- und Kleinschreibung zu normalisieren und Interpunktions- oder Leerzeichen beim Vergleich zu ignorieren. Zudem besteht die Möglichkeit, normalisierte Versionen der Textfassungen zu importieren sowie einen Schwellwert für die intern beim Vergleich zweier Token verwendete Levenshtein-Distanz zu setzen und so einen gewissen Grad an Unschärfe beim Vergleich zu erlauben [38].
Bei TRAViz kann ebenfalls ein Grenzwert für die (relative) Levenshtein-Distanz gesetzt werden [75]. Darüber hinaus werden Textvarianten aufgrund von unterschiedlicher Groß-/Kleinschreibung und Sonderzeichen mit der Standardeinstellung ignoriert.
Der Variance-Viewer hebt sich von den anderen Werkzeugen durch die Möglichkeit ab, über eine anpassbare Konfigurationsdatei[15] eigene Kategorien von Varianten anzulegen und damit deren Normalisierung beim Vergleich zu spezifizieren, insofern die mitgelieferten Standardeinstellungen unzureichend sind. Diese Konfiguration erlaubt beispielsweise projektspezifische Definitionen für Interpunktionen, Grapheme, Abkürzungen sowie Typografie, das Erkennen von Leerzeichen im Wort oder auch das Ignorieren mit einer geringen Editierdistanz [13].
Beim Werkzeug LAKomp wurde bei der Entwicklung ein besonderer Fokus auf die Normalisierung gelegt, um dem untersuchten Werk, einem handschriftlichen

[15] Für ein Beispiel siehe die Dokumentation unter `https://github.com/cs6-uniwue/Variance-Viewer#configuration` (abgerufen am 21.11.2023).

frühneuhochdeutschen Text, gerecht zu werden [84]. Der fehlenden Orthographie und damit sehr unterschiedlichen Schreibweisen des selben Wortes wurde mit der Möglichkeit zur semi-automatischen Annotation begegnet, welche das Lemma, die Wortart und verschiedene morphologische Informationen umfasst. Zur Unterstützung der Nutzer:innen nutzt LAKomp Wörterbücher und lernt mit, sodass es zunehmend besser werdende Vorschläge für die Annotation liefern kann. Der eigentliche Textvergleich findet schließlich auf Basis der annotierten Textfassungen statt, was den Nutzer:innen erlaubt, verschiedene Arten von Textvarianten bei Bedarf auszublenden.

Umgang mit Transpositionen (III-b)

III. (b) Die Änderung der Reihenfolge von Wörtern oder längerer Textsegmente wird als Transposition bezeichnet und kann auf verschiedenen Vergleichsebenen auftreten. Im Folgenden wird zwischen lokalen und globalen Transpositionen unterschieden. Erstere treten auf Wortebene innerhalb eines Textsegments auf, beispielsweise wenn der Satzbau verändert wird. Mit globalen Transpositionen sind hingegen Umsortierungen ganzer Textsegmente in Folge umfangreicher Überarbeitungen eines Werks gemeint, was zu strukturellen Unterschieden zwischen den Textfassungen führt.
Transpositionen von zwei benachbarten Token als mögliche Operation beim Editieren wurden bereits 1964 mit der Damerau–Levenshtein-Distanz [34] modelliert. Beim Vergleich unterschiedlicher Textfassungen ist diese Modellierung allerdings unzureichend, da Transpositionen nur selten zwei direkt benachbarte Wörter betreffen und zudem auch mehrere Wörter gleichzeitig umfassen können, die wiederum untereinander die Reihenfolge tauschen können. So macht das Hinzufügen dieser komplexen Transpositionen zur Menge der üblicherweise modellierten Operationen das Bestimmen der Editierdistanz zwischen zwei Zeichenketten algorithmisch deutlich schwerer, wie Schmidt in [135] für verschiedene Varianten des Problems ausführt. Hinzu kommt das Problem, dass es mitunter eine Interpretationsfrage für Editor:innen ist, ob es sich bei einer Positionsänderung tatsächlich um eine bewusst durchgeführte Transposition oder eine Kombination aus Einfügungen und Löschungen handelt[16], insbesondere bei ohnehin häufig auftretenden Wörtern.

[16] Siehe zum Beispiel in [37, S.3]: „Whether these two edit operations actually can be interpreted as a transposition though ultimately depends on the judgement of the editor and can at best be suggested but not determined via current heuristics“.

Transpositionen werden nur in wenigen der untersuchten Werkzeuge zur Kollationierung berücksichtigt. So findet sich in den Quellen zu TUSTEP, LAKomp[17] und Juxta[18] keine Erwähnung entsprechender Funktionalitäten. Für den Variance-Viewer wird die Erkennung und Visualisierung von Transpositionen hingegen als mögliche zukünftige Erweiterung explizit genannt [13]. Im Werkzeug eComparatio wurde diese Funktionalität prinzipiell implementiert [16], liefert mitunter aber inkorrekte Ergebnisse, wie das in Abbildung 3.8 illustrierte Beispiel zeigt.

Abbildung 3.8: Ein mittels eComparatio verglichener Satz in zwei Fassungen, welche sich lediglich in der Reihenfolge ihrer Wörter unterscheiden. Statt Transpositionen zu erkennen (*VERT* beziehungsweise *VERD*), werden die Textvarianten allerdings als Ersetzungen und Einfügungen (*G* und *MIAT*) klassifiziert.

Bei der Entwicklung der 2009 von Schmidt und Colomb vorgestellten Variantengraphen wurden Transpositionen bereits einbezogen [137]. Diese Grundlage wurde im darauf aufbauenden Werkzeug CollateX weitergeführt und so ist es in der Lage, Transpositionen zu bestimmen und zu visualisieren. Dazu wird in den intern als Datenstruktur verwendeten Variantengraphen ein zusätzlicher Pfad durch Wiederholung einer Knotenfolge erzeugt, wenn eine Transposition durch einen der wählbaren Vergleichsalgorithmen[19] erkannt wurde [36]. Sie werden in der anschließenden Vi-

[17] Auch wenn die Transpositionen nicht angezeigt werden, ist der in LAKomp verwendete Vergleichsalgorithmus, der ebenfalls in LERA zur Alignierung von Textsegmenten genutzt wird, prinzipiell in der Lage, diese zu erkennen. Siehe Abschnitt 5.3.2.

[18] Siehe beispielsweise auch [66, S.41f.], wo anhand eines Beispiels die fehlende Erkennung einer Transposition in Juxta aufgezeigt wird.

[19] Zwei von drei in CollateX verfügbaren Algorithmen erlauben dies.

sualisierung des Graphen durch eine grau gestrichelte Linie angezeigt. Auch TRAViz ermöglicht diese Form der Darstellung, wie anhand eines Beispiels in Abbildung 3.9 illustriert wird. Beide Werkzeuge halten die Behandlung von Transpositionen dabei optional.

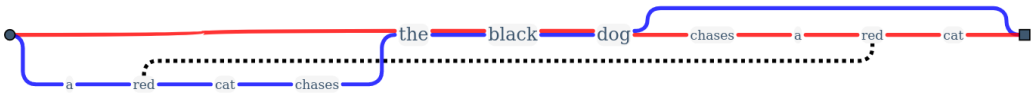


Abbildung 3.9: Das Werkzeug TRAViz erkennt Transpositionen und stellt diese durch gestrichelte Linien im Variantengraphen dar, wenn ein entsprechendes Wort (hier „red“) mit der Maus fokussiert wird.

Neben Variantengraphen eignen sich insbesondere auch alignierte Barcodes für die Darstellung von Transpositionen zwischen genau zwei Textfassungen, die darin durch sich überkreuzende Verbindungen leicht erkennbar sind. Ein Beispiel hierfür ist das Werkzeug iteal (siehe Abbildung 3.7). Prinzipiell können beide Visualisierungstechniken sowohl für lokale (auf Wortebene) als auch für globale (auf Segmentebene) Transpositionen genutzt werden.

Typografie, Semantik und Mehrsprachigkeit (III-c)

Die typografischen Elemente der Textfassungen bleiben von den untersuchten digitalen Werkzeugen mit nur zwei Ausnahmen unberücksichtigt. TUSTEP, als eine der beiden, ermöglicht zwar keinen Vergleich, aber mit dem integrierten Satzprogramm zumindest die Darstellung vieler typografischer Elemente. Dies umfasst Fettungen, Kursivierungen, Sperrungen, Kapitälchen, diverse Formen von Durch-, Über- und Unterstreichungen, farbliche Unterlegungen sowie Hoch- und Tiefstellungen [133]. Der Variance-Viewer ermöglicht darüber hinausgehend die Bestimmung von typografischen Textvarianten. Das Werkzeug wertet dazu die im XML-Attribut *rend* codierten Informationen zur Gestaltung der Texte aus und zeigt die Unterschiede optional an [13]. Die Nutzung des XML-Attributs erlaubt dabei die uneingeschränkte Beschreibung typografischer Informationen. Auch lässt sich in einer Konfigurationsdatei angegeben, wie die Elemente dargestellt werden sollen. Dazu müssen für die möglichen Werte im *rend*-Attribut CSS-Anweisungen (Cascading Style Sheets) hinterlegt werden, die wiederum die Darstellung im Browser spezifizieren.

Nicht alle Textvarianten führen zu einer Bedeutungsverschiebung. Eine semantische Analyse ermöglicht es, dies zu erkennen und so theoretisch inhaltlich übereinstimmende Textpassagen mit unterschiedlicher Wortwahl als Textvariante zu ignorieren, was wiederum die Untersuchung inhaltlicher Änderungen vereinfacht. Momentan unterstützt keines der untersuchten Kollationierungswerkzeuge das Erkennen semantischer Textvarianten, doch handelt es sich bei der semantischen Analyse um ein aktives Forschungsfeld im Natural Language Processing (NLP) mit vielfältigen Ansätzen und Verfahren, welches hier nur kurz angerissen wird. So kommen auf Wortebene, beispielsweise zur Erkennung von Synonymen, einfache Lexika, Wortnetze [93], welche verschiedene semantische Relationen zwischen Wörtern modellieren, oder Worteinbettungen [8], welche Wörter anhand ihrer statistisch geschätzten Bedeutungsähnlichkeit auf Grundlage der sie verwendenden Kontexte in einem hoch-dimensionalen Raum anordnen, zum Einsatz. Der Ansatz der Einbettung kann auch auf ganze Wortgruppen oder Sätze übertragen werden, indem die Werte der Worteinbettung kombiniert werden, beispielsweise mit Distanzfunktionen wie der Word Mover's Distance [83], oder direkt eine Satzeinbettung [86] trainiert wird. Zudem wird aktiv an komplexen Sprachmodellen geforscht, welche für eine Vielzahl von Aufgaben einsetzbar sind. Ein aktuelles, prominentes Beispiel hierfür ist BERT [39], welches auf Basis mehrschichtiger bidirektionaler Transformer arbeitet.

Eng verwandt mit der Erkennung semantischer Übereinstimmungen ist auch der Vergleich von Textfassungen in unterschiedlichen Sprachen. Im Kontext der Editionsphilologie ist die computergestützte sprachübergreifende Analyse noch weitestgehend unbetrachtet. Eine Ausnahme hiervon ist beispielsweise eine Methode zur automatischen Alignierung der portugiesischen und spanischen Textfassungen des *Manual de confesores y penitentes* mittels sprach- und textspezifischer Regeln sowie n-Grammen [148].

In anderen Domänen finden sich hingegen bereits vielfältige Methoden, die einen sprachübergreifenden Vergleich von Texten ermöglichen. Erste Ansätze zur zeilenweisen Alignierung beispielsweise nutzen die Anzahl der Wörter [24] oder basieren auf einer Wort-zu-Wort Übersetzung [26], sodass sie nur für Textfassungen in Form sehr wortgetreuer Übersetzungen infrage kämen. Demgegenüber stehen allerdings Editionsvorhaben, deren Textfassungen nicht nur reine Übersetzungen beinhalten,

sondern auch umfangreiche Überarbeitung der Werke während der Übersetzung in andere Sprachen, was durch die Autor:innen selbst (siehe Beispiele 2.1.4 und 2.1.5) oder durch Dritte (siehe Beispiel 2.1.3) geschehen kann. In Folge dessen ist für die Kollationierung mehr nötig, als die Alignierung übersetzter Textsegmente. Viele State-of-the-Art-Methoden aus der NLP-Forschung nutzen auch für sprachübergreifende Aufgaben die bereits oben erwähnten Einbettungen sowie semantische Sprachmodelle. Ansätze zur sprachübergreifenden Einbettung existieren dabei sowohl auf Wortebene [131] als auch auf Satzebene. So wurden in einer aktuellen Arbeit zunächst monolinguale Satzeinbettungen trainiert und anschließend zu einem gemeinsamen Modell kombiniert [59]. Auch Sprachmodelle wie BERT können für das Trainieren multilingualer Satzeinbettungen genutzt werden [44].

3.2.4 Nutzerfreundlichkeit des Systems

Die Nutzerfreundlichkeit – eigentlich Gebrauchstauglichkeit (engl. Usability) – einer Software ist ein Teilaspekt des gesamten Nutzererlebnisses (engl. User Experience). Bei ihrer Bewertung handelt es sich um ein breites Feld, welches im Rahmen dieser Dissertation nur angeschnitten wird. Doch ist die Nutzerfreundlichkeit im Kontext der Digital Humanities wichtig, um die Akzeptanz einer Software bei den potenziellen Anwender:innen aus den Geisteswissenschaften zu steigern, was insbesondere auch das Niedrighalten der Hürden zur Nutzung der Software mit einschließt [145]. Im Folgenden werden daher drei Kriterien bestehender Werkzeuge untersucht. Zum Ersten: Was müssen Anwender:innen tun, um die Software zu nutzen (Installation)? Zum Zweiten: Wie wird die Software bedient (Nutzerführung)? Und zum Dritten: Wie kompatibel ist die Software mit Standards und anderen Werkzeugen aus der Domäne der digitalen Editionswissenschaften (Interoperabilität)?

Installation (IV-a)

Eine mögliche Hürde für Anwender:innen kann bereits eine notwendige Installation von Software auf dem eigenen System sein. Viele der untersuchten Werkzeuge setzen diese sowie ein gewisses technisches Wissen dafür voraus. Dazu gehört TUSTEP, welches für verschiedene Betriebssysteme verfügbar ist und eine ausführliche Installationsanleitung bereit stellt[20]. Dabei ist der Umgang mit Kommando-

[20] Abrufbar unter `https://www.tustep.uni-tuebingen.de/inst.html` (besucht am 28.11.2023).

zeilenwerkzeugen sowie deren Konfiguration notwendig. Auch andere Werkzeuge, wie CollateX, Juxta – sowohl in der ursprünglichen Desktop-Version als auch in der überarbeiteten Form als *Juxta Web Service* – und Variance-Viewer werden mittels Kommandozeile installiert. Die drei Werkzeuge basieren auf Java und benötigen zur Ausführung eine Java Runtime Environment (JRE). Diese Abhängigkeiten zu anderen Software-Paketen bringen wiederum weitere Hürden mit sich. Zum einen werden mitunter Administrationsrechte benötigt. Zum anderen kann die Verfügbarkeit der benötigten Software-Pakete eingeschränkt sein[21] oder die mitgelieferten Installationsanleitungen können veralten[22].

Einige Tools implementieren einen webbasierten Ansatz, beispielsweise Variance-Viewer oder eComparatio, was für die Ausführung auf dem eigenen System das Wissen zum Starten eines lokalen Webservers voraussetzt. Über die damit einhergehende Server-Client-Architektur ließe sich die Installation vor Endnutzer:innen aber prinzipiell verbergen, indem die Einrichtung und Wartung des notwendigen Servers an Dritte übergeben und die Anwendung damit bequem im Browser ohne eigene Installation aufrufbar wird. So können beispielsweise eComparatio[23] und Stemmaweb[24] ohne Installation auf dafür eingerichteten Seiten genutzt werden. Auch mit Juxta Commons [153], welches Juxta registrierten Nutzer:innen als Webapplikation zur Verfügung stellte, im September 2020 allerdings abgeschaltet wurde [80], wurde dieser Ansatz der niederschwelligen Nutzbarkeit der Software realisiert. Für CollateX ist ein Demonstrator mit grafischer Nutzeroberfläche frei im Internet verfügbar[25], mit welchem eigene Beispiele ganz ohne Registrierung kollationiert werden können.

[21] Zwar ist der Quellcode von Juxta weiterhin unter `https://github.com/performant-software/juxta-service` (besucht am 12.05.2022) verfügbar, doch lässt sich die Software mit aktuellen Versionen des Java Runtime Environments nicht mehr bauen. Zudem sind einige der benötigten Ressourcen nicht mehr verfügbar, die von `https://gregor.middell.net/maven/content/repositories` abgerufen werden.

[22] Beispielsweise wird in der Installationsanleitung des Variance-Viewers für Linux das Paket „tomcat8“ gefordert, welches allerdings in den aktuellen Paketquellen nur noch in einer neueren Version verfügbar ist.

[23] Verfügbar unter `http://www.ecomparatio.net/` (besucht am 1.12.2023).

[24] Verfügbar unter `https://stemmaweb.net/stemmaweb/` (besucht am 6.12.2023), Registrierung erforderlich.

[25] Verfügbar unter `https://collatex.net/demo/` (besucht am 1.12.2023).

Nutzerführung (IV-b)

Ein weiterer wichtiger Aspekt, der hier kurz andiskutiert wird, ist die eigentliche Bedienung der untersuchten Werkzeuge.
CollateX wird abseits des genannten Demonstrators über die Kommandozeile bedient. Auch TUSTEP ist als Kommandozeilenwerkzeug implementiert und wird mittels eines eigenen, umfangreichen Befehlssatzes bedient, welcher einer Programmiersprache gleicht. Abbildung 3.10 zeigt ein kurzes Beispiel. Der große Funktions-

```
#DATEI, txt, SEQ-P #EDIERE, txt, T
#FORMATIERE, txt, LOESCHEN=+, PARAMETER=*
DRT        HPLJ
KT         / @z - &#3 - /
*EOF
#DRUCKE, , HPLJ
```

Abbildung 3.10: Ein aus [133, S.111] übernommenes Beispiel für die Bedienung von TUSTEP. Mit den Kommandos wird eine Datei eingerichtet, bearbeitet und derart formatiert, dass sie mit dem Zeichenvorrat des Druckers kompatibel ist sowie Seitennummern beinhaltet, bevor sie schließlich gedruckt wird.

umfang von TUSTEP, der alle Schritte die Edierens von der Transkription über den Textvergleich bis hin zum Buchsatz abdeckt, führt zu einer komplexen Steuerung des Systems. So besitzt das offizielle Handbuch in der Version von 2022, siehe [133], 1.453 Seiten. Ein flüssiges Arbeiten setzt entsprechend eine längere Einarbeitung und Erfahrung mit dem Werkzeug voraus. Mit TXSTEP ist eine XML-Schnittstelle in Entwicklung, mit welcher TUSTEP in XML-Syntax programmiert werden kann, sodass neben der Nutzung einer verbreiteten Beschreibungssprache auch die Vorteile von XML-Editoren, wie Autovervollständigung und Syntax-Highlighting, die Nutzerfreundlichkeit verbessern [110].
Die meisten der untersuchten Werkzeuge bieten eine grafische Nutzeroberfläche an, sobald die Installation abgeschlossen ist. Dazu gehören beispielsweise Juxta, Variance-Viewer, eComparatio oder TRAViz, wobei letzteres als visueller Bestandteil einer Webseite konzipiert ist, sodass zur Nutzung zunächst Daten, Parameter und einige Befehlen via JavaScript zu spezifizieren sind. Grafische Nutzeroberflächen bieten Nutzer:innen ohne technischen Hintergrund einen leichteren Einstieg in die

Software als die Bedienung via Kommandozeile oder mittels Programmierung. Wie zugänglich diese Nutzeroberflächen sind, ist in den meisten Fällen allerdings fraglich, da ihre Nutzerfreundlichkeit nur selten mittels Nutzerstudien evaluiert wird. Eine Ausnahme hiervon bildet TexTile, für welches eine Studie mit 15 Teilnehmer:innen durchgeführt wurde [10]. Anhand 13 gestellter Aufgaben, wie dem Bestimmen der Textvarianten für eine vorgegebene Stelle, und 10 qualitativer Fragen, wie der Einschätzung nach der Nützlichkeit bestimmter Visualisierungen zum Abgleich der Varianten, konnten die Stärken und Schwächen der Software identifiziert sowie Ideen für zukünftige Funktionalitäten abgeleitet werden.
Fast alle untersuchten Werkzeugen bieten Nutzer:innen darüber hinaus Dokumentationen, Nutzerhandbücher oder auch Online-Tutorials an, was den Einstieg und Umgang mit der Software erleichtert.

Interoperabilität (IV-c)

Zur Nutzerfreundlichkeit gehört schließlich auch die Frage, ob Textfassungen unkompliziert in die Software importiert und Kollationierungsergebnisse gegebenenfalls zur Weiterverarbeitung exportiert werden können. Das Unterstützen von etablierten Austauschformaten sowie Bereitstellen von Programmierschnittstellen erhöht dabei zudem die Interoperabilität eines Werkszeugs und vereinfacht es, dieses in einen mehrteiligen Arbeitsablauf zu integrieren.
Ein verbreitetes Austauschformat für verschiedene Daten im Kontext digitaler Editionen ist Extensible Markup Language (XML). Insbesondere XML nach den Richtlinien der *Text Encoding Initiative* (TEI) [29], welche sich seit ihren Ursprüngen 1987 vor allem um Leitlinien für die Kodierung maschinenlesbarer Texte in den Geistes- und Sozialwissenschaften sowie der Linguistik verdient gemacht hat, gilt dabei als de-facto-Standard. So wird XML von vielen der untersuchten Werkzeuge, beispielsweise CollateX, Juxta oder Variance-Viewer, als Importformat unterstützt. Weitere verbreitete Importformate sind JSON (JavaScript Object Notation), welches beispielsweise von CollateX und TRAViz unterstützt wird, und einfache Textdateien, welche von Werkzeugen wie Juxta, LAKomp, TUSTEP und Variance-Viewer verarbeitet werden können. LAKomp weist hierbei die Besonderheit auf, dass es eine spezielle Transkriptionsnotation für die Beschreibung frühneuhochdeutscher Texte verarbeitet, welche an einem Beispiel in Abbildung 3.11 illustriert wird.
Während die Menge der unterstützten Eingabeformate der Werkzeuge relativ ähnlich ist, unterscheidet sie sich beim Export je nach Art des Ergebnisses und dem

```
Bu-86r25    +Ü pfeyl aus*2a zu zcy\:enn*1a(.) @Ü
Bu-86r26    *f*(I*)tm\-(,) nu\* wil ich dich lernen(,) wy\: du\*
Bu-86v01    eym dy\: pfeyl au\*s*2a czihnn\-*1a $zalt der ge$cho$$znn\_
Bu-86v02    i$t ader mit was czeuges an czangen(.)
Bu-86v03    vnde dir $zu\*$t etlich lere geben(,) wy#lang
Bu-86v04    du\* den pfeyl $chtecken le$$znn\- $zalt vn\-d
Bu-86v05    woru\*m\-b(.) das finde$zt|u\* hirnoch ge$chribenn\_(.)§ +K ge$chribenn\_(.): AbsLZ @K
Bu-86v06    *f*{I*}tm\-(,) kompt*1a dir einer czu\*§*2a vn\-d i$t du\*rch
Bu-86v07    den leip ge$cho$$zen(,)§ alzo durch dy hole
Bu-86v08    au\*s(,)§ zo rot ich dir mit#nichtnn\-(,)§ da$z|tu\*
Bu-86v09    ym den pfeyl aus*2b czeu\*e$t*1b(,)§ doru\*mb(,) wen
Bu-86v10    du\* yn zo balde au\*s*2c|czoge$t*1c(,)§ $zo lyff das
Bu-86v11    blu\*t al#vm\-b von allen enden zu der
Bu-86v12    wu\*nden(.)§ i$t er gantz durch den leip
Bu-86v13    ge$cho$$zen(,) zo leu\*fft*1a das blu\*t hinden
Bu-86v14    vnde forn zu\* den lochern aus*2a als eyn
```

Abbildung 3.11: Im Werkzeug LAKomp werden Textfassungen als einfache Textdatei in einer speziellen Transkriptionsnotation eingelesen und verarbeitet. Das gezeigte Beispiel wurde von `https://lakomp.uzi.uni-halle.de/` übernommen und findet sich in ähnlicher Form in [84].

möglichen Verwendungszweck. So bieten manche Tools fertige Lese-Darstellungen des Textvergleichs an, beispielsweise im Falle des Variance-Viewers in Form einer PDF-Datei. Dazu gehören auch HTML-Formate ganzer Synopsen (eComparatio, Juxta) oder einzelner Visualisierungen (TRAViz), welche wiederum als ein Bestandteil in digitale Editionen eingebunden werden können. Auch die von CollateX erstellten Variantengraphen können so im GRAPHML- oder DOT-Format weiterverwendet werden. Daneben bieten beispielsweise CollateX, eComparatio und Variance-Viewer mit TEI-XML und JSON zwei für die maschinelle Weiterverarbeitung gedachte Formate für den Download an.

Maschinenlesbare Formate erlauben die Verknüpfung mehrerer Werkzeuge und steigern damit deren Interoperabilität. Tatsächlich gibt es einige Werkzeuge, welche primär der (interaktiven) Darstellung kollationierter Textfassungen dienen, ohne selbst eine Kollationierung zu berechnen. Ein prominentes Beispiel hierfür ist die Versioning Machine [138, 14], welche als Importschnittstelle ein XML-Format nach den Richtlinien der Text Encoding Initiative (TEI) definiert, die wiederum von Tools wie Juxta als Exportoption angeboten wird. Weitere Beispiele sind das Visualisierungswerkzeug PyCoviz [107], das die von CollateX bereitgestellten JSON-Daten verarbeiten kann, und Stemmaweb, in welches sich unter anderem GRAPHML-Daten aus CollateX importieren lassen. Juxta und CollateX können wiederum genutzt werden, um auf anderem Wege erlangte Kollationierungsergebnisse zu visualisieren.

Einige Werkzeuge bieten neben den manuellen Download-Optionen auch Programmierschnittstellen (APIs) an, auf welche direkt maschinell zugegriffen werden kann, um eine automatisierte Vernetzung verschiedener Software zu ermöglichen. Dazu gehören sowohl CollateX als auch Juxta, welche beide eine auf JSON als Austauschformat ausgelegte RESTful-API anbieten.

3.2.5 Interaktivität des Systems

Im Folgenden wird die Interaktivität der grafischen Nutzeroberflächen und Visualisierungen einiger der bereits vorgestellten Ansätze untersucht. Zwar können auch Kommandozeilenwerkzeuge zu einem gewissen Grad interaktiv gestaltet werden, doch ist dies für die weitere Dissertation irrelevant und wird hier entsprechend ausgespart, da aufgrund der angestrebten Nutzerfreundlichkeit ein Werkzeug mit grafischer Nutzeroberfläche entwickelt wurde. Die nachfolgende Untersuchung gliedert sich in Aspekte der interaktiven Bestimmung und der interaktiven Analyse der Textvarianten. Eine ähnliche Zweiteilung existiert in [156], wo neben einer umfangreichen Übersicht der unterschiedlichen Ansätze zur Visualisierung von Text-Alignierungen auch deren verschiedene Interaktionsmöglichkeiten zusammenfassend aufgelistet werden.

Interaktive Bestimmung von Textvarianten (V-a)

Viele Werkzeuge gestalten die grundlegende Wahl der zu vergleichenden Textfassungen interaktiv über die Nutzeroberfläche. In Textile existiert dafür beispielsweise eine Liste aller verfügbaren Textfassungen, aus denen die darzustellende Teilmenge gewählt werden kann. Bei nach dem Leitquellenprinzip arbeitenden Kollationierungswerkzeugen, wie Juxta oder eComparatio, lässt sich zudem der Basistext interaktiv wählen, was wiederum Einfluss auf die Textvarianten hat und so eine Neuberechnung erforderlich machen kann.
Die Normalisierung der Textfassungen beziehungsweise Filterung der anzuzeigenden Textvarianten ist ebenfalls häufig interaktiv über die Nutzeroberfläche steuerbar. So kann beispielsweise im Variance-Viewer das Hervorheben verschiedener Textvarianten, wie Graphemen, Satzzeichen und Abkürzungen, entsprechend der zuvor bestimmten Klassifikation jederzeit an- und abgeschaltet werden [129]. Auch bei LAKomp werden die detaillierten Annotationen genutzt, um verschiedene Arten von Textvarianten interaktiv ein- und auszublenden. Andere Werkzeuge, wie TRAViz,

erlauben mittels interaktiver Schieberegler einen Schwellwert für die Editierdistanz festzulegen, bei dessen Unterschreitung zwei Token noch als identisch angesehen werden.
Eine weitere, von einigen Werkzeugen realisierte Interaktionsmöglichkeit stellt ein nachträglicher Eingriff in automatisch bestimmte Kollationierungsergebnisse dar, sodass Textvarianten in einem semi-automatischen Prozess von Nutzer:innen festgelegt werden können. Möglich ist dies beispielsweise in Juxtas *Side-by-Side*-Visualisierung und in den von TRAViz generierten Variantengraphen, welche ein interaktives Auftrennen oder Verschmelzen von Knoten ermöglichen. In [13] wird eine entsprechende Editierfunktion des automatisch generierten Ergebnisses als mögliche Erweiterung für den Variance-Viewer genannt.

Interaktive Analyse der Textvarianten (V-b)

Eine mögliche Form für digitale Werkzeuge, die Analyse von Textvarianten gerade in sehr umfangreichen Werken zu unterstützen, ist eine integrierte Suchfunktion. Diese interaktive Funktionalität findet sich in vielen digitalen Editionen, aber nur wenigen Kollationierungswerkzeugen, wie Juxta und LAKomp.
Hingegen besitzen die meisten Werkzeuge interaktive Funktionalitäten zum Einblenden zusätzlicher Informationen zu den Textvarianten in Form von Popups, die beim Klick auf bestimmte Elemente oder beim Berühren mit der Maus (Hover-Effekt) eingeblendet werden. Beispielsweise nutzen eComparatio, Juxta und TRAViz die Möglichkeit zum optionalen Einblenden von Variantenapparaten, die an entsprechenden Stellen in ihren Volltextdarstellungen hinterlegt sind. In iteal werden gar drei Visualisierungsformen für verschiedene Vergleichsebenen (Textfassungs-, Vers- und Wortebene) per Mausklick miteinander verknüpft, um so ausgehend vom gesamten Text immer detaillierter werdende Informationen anzuzeigen (siehe Abbildung 3.7).
Diese Formen der Interaktion können ebenfalls zur visuellen Hervorhebung verknüpfter Elemente genutzt werden. Ein Beispiel hierfür ist der Maus-Hover-Effekt für die Knoten (Repräsentanten für Token) in den von TRAViz generierten Variantengraphen, welcher die Pfade aller dazugehörigen Textfassungen hervorhebt, während alle anderen Kanten ausgeblendet werden.
Interaktive Elemente werden auch zur Navigation in Visualisierungen genutzt. Zum einen sind sie nützlich um den angezeigten Bildschirmausschnitt durch Verschiebung zu ändern, wenn die Visualisierung über die Größe des Bildschirms übersteigt.

Das wird beispielsweise in Stemmaweb auf zwei Arten umgesetzt: in den Variantengraphen mittels Scrollleiste und in den Stemmagraphen mittels einer Kombination der Mausinteraktionen Klicken, Halten und Bewegen (Panning). Zum anderen erlaubt eine integrierte Zoomfunktion den Zugriff auf verschiedene Detailstufen, wenn gleichzeitig sehr viele Informationen innerhalb einer Visualisierung dargestellt werden sollen. Auch diese interaktive Funktionalität setzt Stemmaweb auf zwei unterschiedliche Arten um: in den Variantengraphen mittels eines Schiebereglers und in den Bäumen mittels des Mausrads beziehungsweise einer entsprechenden Zoomgeste bei berührungsempfindlichen Eingabegeräten.

3.2.6 Generizität des Systems

Lösungen für die vier Zielsetzungen der Kollationierung (VI-a)

In Abschnitt 2.2.3 des vorherigen Kapitels wurden vier mögliche Zielsetzungen der Kollationierung beschrieben. Zwei von diesen, die Unterstützung der Transkription von Textfassungen sowie die Dokumentation von Textvarianten zwischen den Textfassungen, werden mithilfe von Kollationierungssoftware durch das Anzeigen der Textvarianten direkt realisiert. Zudem sind bei der Dokumentation die (teils dynamisch) verfügbaren Normalisierungsoptionen hilfreich, um wiederkehrende oder entsprechend der getroffenen Editionsrichtlinien nicht zu dokumentierende Klassen von Textvarianten, zum Beispiel Unterschiede in der Groß-/Kleinschreibung oder Interpunktion, vom Vergleich auszuschließen. Die Editionsrichtlinien bleiben damit zudem bis zu einem gewissen Grad dynamisch anpassbar.
Auch zur Untersuchung der Filiation, das heißt des Abstammungsverhältnisses der Textfassungen, als die dritte der genannten Zielsetzungen gibt es unterstützende Ansätze. Neben dem reinen Aufzeigen von Einzelstellen, können die berechneten Textvarianten in geeigneter Form aggegriert werden, um Ähnlichkeiten auf Fassungsebene übersichtlich zu visualisieren. Dafür können die bereits genannten phylogenetischen Bäume genutzt werden [68], welche in adaptierter Form als Stemmagraphen in Stemmaweb implementiert wurden. Eine alternative Visualisierungsform für den selben Zweck findet sich in LAKomp in Form von Ähnlichkeits- beziehungsweise Konfliktgraphen. Darin werden die Textfassungen als Knoten dargestellt, welche jeweils paarweise durch eine Kante verbunden werden. Das Layout des Graphen ergibt sich über ein Federspannmodell anhand der Kantengewichte, welche wiederum anhand der Kollationierung der Textfassungen auf Wortebene abgeleitet werden.

Dabei ist eine Wichtung für verschiedene Wortarten möglich. Abbildung 3.12 zeigt einen solchen Ähnlichkeitsgraphen am Beispiel.

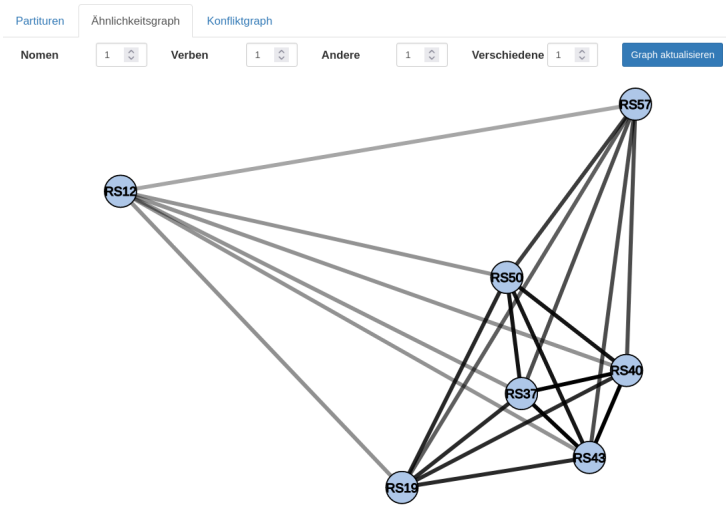


Abbildung 3.12: Ein mittels LAKomp generierter Ähnlichkeitsgraph für das Märchen *Rumpelstilzchen* in sieben Textfassungen.

Bei beiden Visualierungsformen ist die Wichtung der einzelnen (Klassen von) Textvarianten entscheidend für das Ergebnis. Da dies wiederum schwer objektiv fassbar sowie eine generelle Wichtung für verschiedene Textgattungen nicht sinnvoll ist, sehen auch die Entwickler:innen von Stemmaweb diesen aggregierenden Ansatz prinzipiell kritisch [9]. Sie versuchen, sich dem Problem dennoch in [9] mit einem flexiblen Modell zu nähern, welches möglichst viele Formen von Textvarianten berücksichtigt.
Für die vierte Zielsetzung laut Abschnitt 2.2.3, der Untersuchung von Textvarianten zur Beantwortung historischer Fragestellungen, sind ebenfalls umfassende Normalisierungsfunktionen hilfreich, sodass sich Forschende auf die inhaltlichen Änderungen konzentrieren können. Dies ist beispielsweise in LAKomp durch die umfangreiche Annotation der Textfassungen möglich, welche das Ausblenden graphematischer, morphologischer und lexikalischer Varianten erlaubt [84].

Unterstützte Sprachen (VI-b)

Wesentlich für die Generizität einer Software in der Domäne der digitalen Editionsphilologie ist die Vielfalt der Sprachen, welche von der Software unterstützt werden.

Sprachen mit nicht-lateinischen Alphabeten stellten früher oft ein Problem dar, beispielsweise aufgrund des beschränkten Zeichensatzes in der verbreiteten ASCII-Kodierung. TUSTEP begegnete diesem Problem mit einer eigenen Form der Kodierung in lateinischer Umschrift. So berücksichtigt dessen *Zeichenvorrat* verschiedene Sonderzeichen und nicht-lateinische Alphabete wie Griechisch, Hebräisch oder Arabisch, und wurde vielfach für Editionsprojekte entsprechender Werke verwendet[26]. In neueren Werkzeugen, wie Juxta und CollateX, basieren die unterstützten Importformate sowie die interne Kodierung der Texte häufig auf UTF-8, sodass die Verarbeitung nicht-lateinischer Alphabete sowie der meisten Sonderzeichen unproblematisch ist. Die korrekte Darstellung ist allerdings von der verwendeten Schriftart abhängig, welche die entsprechenden Zeichen unterstützen muss und mitunter vom System der Nutzer:innen abhängt.

Bei der Darstellung spielt zudem die Schreibrichtung der Sprache eine wichtige Rolle und führt zu Problemen, wenn das Werkzeug nicht an die verwendete Sprache der Texte angepasst ist. So wird im Online-Demonstrator von CollateX das Hebräische in den Variantengraphen und Alignierungstabellen von links nach rechts dargestellt, was die Wortreihenfolge der Zeugen durcheinander bringt. TRAViz erlaubt hingegen die Umkehrung der Schreibrichtung bei der Visualisierung. Von den untersuchten Methoden, für welche die Darstellung der Schreibrichtung relevant ist, unterstützt wiederum keine eine vertikale, welche beispielsweise für viele asiatische Sprachen wichtig wäre.

Eine Konfigurationsmöglichkeit für die Normalisierung ist ebenfalls hilfreich, um den Besonderheiten unterschiedlicher Sprachen gerecht zu werden. Während es bei den untersuchten Werkzeugen meist vordefinierte, feste Kategorien für die Normalisierung gibt, sticht der Variance-Viewer durch die Möglichkeit zum Einspielen einer Konfigurationsdatei positiv hervor. Damit können eigene Kategorien für das Erkennen und Ausblenden von Textvarianten definiert und so sprachabhängig gestaltet werden.

[26] Für eine umfassende Liste siehe `http://www.tustep.uni-tuebingen.de/ed3.html` (besucht am 27.01.2024).

3.2.7 Zwischenfazit

Dieses Unterkapitel hat anhand vieler Beispiele aufgezeigt, welche bestehenden Lösungen es bereits für die sechs identifizierten Anforderungen an Kollationierungssoftware gibt, aber auch, wo deren Schwächen liegen. So erfüllen viele der untersuchten Werkzeuge bereits die wesentliche Anforderung nicht, mehr als zwei Textfassungen gleichzeitig vergleichen zu können, und wenn dann oft nach dem Leitquellenprinzip. CollateX und TRAViz sind hierbei zwei Ausnahmen und können teilweise mit sehr vielen Textfassungen gleichzeitig umgehen. Beide Ansätze haben wiederum Probleme mit der Verarbeitung und Darstellung umfangreicher Textfassungen. Eine mögliche Lösung stellt ein mehrstufiger Ansatz für die Kollationierung dar, wie sie LAKomp oder auch die Visualisierung iteal realisieren. Zudem verfügt CollateX, wie einige andere Werkzeuge auch, abseits des Demonstrators über keine grafische Nutzeroberfläche, sondern muss via Kommandozeile installiert und bedient werden, was eine mögliche Hürde für Nutzer:innen darstellen kann. Der von einigen Werkzeugen realisierte webbasierte Ansatz, der Nutzer:innen das Werkzeug ohne eigene Installation und mit einer grafischen Nutzeroberfläche im Browser zur Verfügung stellt, ist sinnvoll, um die Nutzerfreundlichkeit zu steigern. Viele Ansätze weisen interaktive Elemente für unterschiedliche Aufgaben auf, doch scheitern mitunter an Aspekten der Generizität, beispielsweise der Handhabung nicht-lateinischer Alphabete oder der Schreibrichtung Rechts-nach-Links.

Kapitel 4

LERA – Ein Werkzeug zur Analyse komplexer Textvarianten

LERA steht als Akronym für *Locate, Explore, Retrace and Apprehend complex text variants* und ist ein digitales Werkzeug zur Berechnung und Analyse von komplexen Varianten zwischen unterschiedlichen Fassungen eines Textes, welches primär für Nutzer:innen aus den Geisteswissenschaften entwickelt wurde. Textvarianten sollen damit leicht lokalisiert und mittels interaktiver Visualisierungen explorativ analysiert werden können, sodass das Nachvollziehen und schließlich das Verstehen der Hintergründe dieser Änderungen unterstützt wird. Verschiedene Aspekte der Software bilden das Thema dieser Dissertation und sind Ergebnisse eigener, langjähriger Forschungs- und Entwicklungstätigkeit in Kooperation mit Kolleg:innen verschiedener Fachdisziplinen.

LERAs Ursprung geht auf das interdisziplinäre Forschungsprojekt *Semi-automatische Differenzanalyse von komplexen Textvarianten (SaDA)* zurück, welches vom Bundesministerium für Bildung und Forschung (BMBF) von 2012 bis 2015 gefördert wurde. Im Rahmen dessen wurde LERA anhand des Fallbeispiels der *Histoire des deux Indes* (siehe Abschnitt 2.1.1) konzipiert und implementiert. Eine von Susanne Schütz vom Institut für Romanistik der Martin-Luther-Universität Halle-Wittenberg erstellte Editionsprobe diente dabei als Blaupause für die grundlegende Darstellung der Vergleichsergebnisse in Form einer absatzweisen Spaltensynopse mit farblichen Hervorhebungen der Textvarianten sowie eines listenförmigen Variantenapparats. Abbildung 4.1 zeigt einen Auszug dieser Editionsprobe.

Die Software findet seither Anwendung in verschiedenen Editionsprojekten welt-

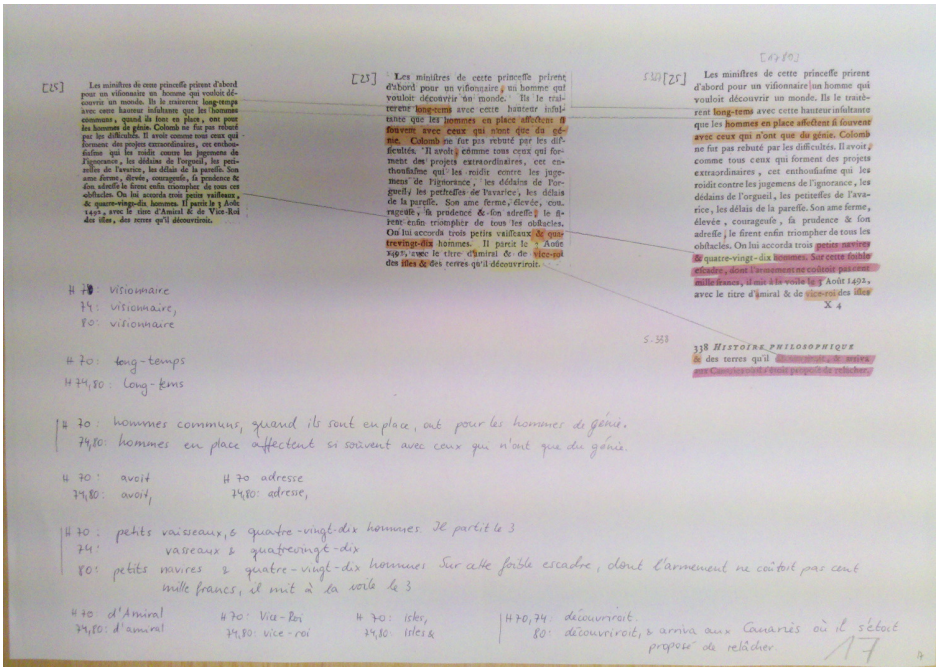


Abbildung 4.1: Ein Auszug aus einer von Susanne Schütz erstellten Editionsprobe der *Histoire des deux Indes*. Dabei wurde ein Absatz der Textfassungen von 1770, 1774 und 1780 spaltensynoptisch angeordnet und die Textvarianten farblich hervorgehoben. Eine Besonderheit stellen die Farbmarkierungen dar, mit denen korrespondierende Textpassagen in allen drei Fassungen hervorgehoben wurden, insofern eine der Fassungen einen Unterschied aufweist. Zudem wurden die Textvarianten in Gänze in einem listenförmigen Variantenapparat wiederholt.

weit, wie im Abschnitt 7.1.2 des Fazits weiter ausgeführt wird. Zuletzt wurde die Weiterentwicklung von LERA für die Erstellung einer hybriden Edition des kabbalistischen Traktats *Keter Shem Ṭov* von der Deutschen Forschungsgemeinschaft (DFG) gefördert (siehe Abschnitt 2.1.2).

Dieses Kapitel widmet sich im Weiteren der Zielsetzung von LERA, dem Abgleich von LERAs Funktionalitäten mit dem Gothenburg-Modell und schließlich der grundlegenden Architektur der Software. Zwei Schwerpunkte der Software, die Berechnung und die interaktive Analyse der Textvarianten in LERA, werden in den beiden nachgelagerten Kapiteln 5 und 6 im Detail diskutiert.

4.1 Zielsetzung

Die Zielsetzung für LERA ist es, ein digitales Werkzeug zu schaffen, welches die in Abschnitt 2.3 aus den verschiedenen Editionsprojekten abgeleiteten Anforderungen an die Kollationierung verschiedener Textfassungen aufgreift und erfüllt. Als Anwendungsdomäne werden dabei die Geisteswissenschaften betrachtet, obgleich die Software letztlich auch für den Textvergleich in anderen Domänen nutzbar ist. Eine Besonderheit ist, dass LERA nicht nur als Hilfsmittel für Editor:innen zur Erstellung einer Edition konzipiert wurde, sondern auch für die spätere Nutzerschaft der digitalen Edition.
Die im Rahmen dieser Dissertation gesteckten und behandelten Ziele an LERA sind:

I. Vergleich von mehreren Textfassungen
   (a) Handhabung von mehr als zwei Textfassungen
   (b) Realisierung des gleichrangigen Vergleichsprinzips
   (c) Umgang mit vielen (Hunderten) Textfassungen

II. Vergleich von umfangreichen Textfassungen
   (a) Direkte Verarbeitung ganzer Bücher
   (b) Realisierung einer mehrstufigen Kollationierung
   (c) Kombination von Close- und Distant-Reading

III. Umgang mit komplexen Textvarianten
   (a) Möglichkeit zum Ausblenden einzelner Klassen von Textvarianten
   (b) Handhabung struktureller Unterschiede zwischen den Textfassungen

IV. Nutzerfreundlichkeit des Systems
   (a) Möglichkeit zur Nutzung ohne Installation
   (b) Realisierung einer möglichst intuitiven Nutzeroberfläche
   (c) Bereitstellung verbreiteter Ein- und Ausgabeformate

V. Interaktivität des Systems
   (a) Flexibilität der Parameterwahl beim Bestimmen von Textvarianten

(b) Möglichkeit zur explorativen Analyse von Textvarianten

(c) Realisierung als Echtzeitanwendung

VI. Generizität des Systems

(a) Lösungen für die vier Zielsetzungen der Kollationierung laut 2.2.3

(b) Unterstützung der Sprachvielfalt

(c) Konfigurierbarkeit des Systems

Hinzu kommen einige weitere Zielsetzungen an die Software, wie beispielsweise die Kollationierung von verschiedensprachigen Textfassungen, die im Ausblick (Abschnitt 7.2) thematisiert werden.

4.2 Abgleich mit dem Gothenburg-Modell

Die Arbeitsweise von LERA adaptiert die im Gothenburg-Modell (siehe 3.1.2) spezifizierten Aufgaben, wie sie beispielsweise in der Dokumentation von CollateX [38] als die fünf aufeinander folgenden Schritte Tokenisierung, Normalisierung, Alignierung, Analyse und Visualisierung aufgeführt werden. In LERA gibt es dabei einige Besonderheiten im Kontrast zu anderen Realisierungen des Gothenburg-Modells bezüglich der Einteilung und Abfolge der Aufgaben. So wird zum einen der Schritt der Tokenisierung in die eigentliche Zerlegung der Eingabedokumente in Token und eine nachgelagerte Segmentierung aufgetrennt. Der Schritt der Normalisierung findet nicht losgelöst von anderen Schritten statt, sondern an mehreren Stellen im System. Der Schritt der Alignierung wird hierarchisch in zwei aufeinander folgenden Stufen durchgeführt: zunächst einer Zuordnung ähnlicher Textsegmente und anschließend einem detaillierten Vergleich dieser aliginerten Segmente auf Tokenebene. Wie die Normalisierung wird schließlich auch die Analyse mit LERA nicht als eigenständiger Schritt ausgewiesen, sondern ist, insbesondere mittels interaktiver Visualisierungen, an verschiedenen Stellen im Ablauf möglich. Ein Abgleich der Arbeitsweise von LERA und dem Gothenburg-Modell findet sich auch in [118], auf dem der gesamte Abschnitt 4.2 beruht.

4.2.1 Tokenisierung und Segmentierung

Die Tokenisierung einer Textfassung in LERA wird in zwei separaten Teilschritten durchgeführt. Der erste Teilschritt ist eine klassische Tokenisierung, bei der eine Textfassung in ihre Token, das heißt ihre kleinsten Komponenten, zerlegt wird. Dieser Prozess findet einmalig beim Upload einer Datei ins System statt und dient als fixe Datengrundlage für die weitere Verarbeitung und Nutzung der Textfassung in LERA. Dabei gibt es 20 verschiedene Token-Klassen mit unterschiedlichen Aufgaben. So existieren unter anderem Klassen für den eigentlichen Inhalt, beispielsweise für Wörter, Zahlen, Satzzeichen und Leerzeichen, aber auch für Strukturelemente, die aus LERA-spezifischen Steuersequenzen und XML-Tags (falls die Textfassung als XML-Datei hochgeladen wurde) extrahiert werden. Zur Identifizierung der einzelnen Token-Klassen wurde ein eigener Tokenizer entwickelt, der mit Mustern auf Basis von regulären Ausdrücken sowie dem XML-Parser Nokogiri [112] arbeitet. Für erkannte XML-Token wird zudem ein Verweis auf das dazugehörige öffnende beziehungsweise schließende XML-Token gespeichert, sodass später effizient auf diese zugegriffen werden kann.
Editionsprojekte nutzen üblicherweise XML-Dateien nach den Richtlinien der Text Encoding Initiative (TEI) als Austauschformat. Allerdings unterscheiden sich zwischen den Projekten häufig die ausgezeichneten Elemente und sogar die verwendeten XML-Tags bei gleichen Elementen. Das folgende Beispiel zeigt dies anhand von Personenauszeichnungen, wie sie unter anderem in den Richtlinien der TEI dokumentiert sind[1] und zum anderen im Editionsvorhaben zum *Manual de confesores* verwendet werden (siehe Abschnitt 2.1.4)[2]:

```
<persName ref=”cerl:cnp00396685 gnd:118622110” key=”Thomas de Aquino”>
  B. Thomas
</persName>

<name type=”person”>Thomas Hoccleve</name>
```

Aus diesem Grund wurde in LERA ein optionaler Vorverarbeitungsschritt vor der eigentlichen Anwendung des Tokenizers integriert, der die Konvertierung der pro-

[1] Das Beispiel stammt aus der Dokumentation zum Element *name*, siehe `https://tei-c.org/release/doc/tei-p5-doc/en/html/ref-name.html` (besucht am 27.01.2024).

[2] Das Editionsprojekt orientiert sich dabei an den Editionsrichtlinien des Projekts *Die Schule von Salamanca*, aus dem auch das Beispiel stammt; siehe `https://www.salamanca.school/de/guidelines.html` (besucht am 27.01.2024).

jektspezifischen XML-Auszeichnungen mittels eines austauschbaren XSLT-Skripts (Extensible Stylesheet Language Transformation) in für LERA verarbeitbare XML-Auszeichnungen ermöglicht.[3].

Im zweiten Teilschritt werden die Token zu größeren Textsegmenten, wie Absätze, Zeilen, Sätze oder Teilsätze, zusammengefasst. Die Segmentierungseinheit kann dabei sowohl initial beim Upload als auch nachträglich über LERAs grafische Nutzeroberfläche festgelegt werden. Die Bereitstellung dieser Funktionalität ist auch der Hintergrund der Zweiteilung des Tokenisierungsprozesses: Die Segmentierung kann einfach angepasst werden, ohne dass die zugrunde liegende Zerlegung in Token erneut berechnet werden muss. Je nach gewählter Einheit erfolgt die Segmentierung an bestimmten Zeichen, wie Leerzeichen oder Interpunktionen, oder an bestimmten XML-Tags, die während der Tokenisierung identifiziert wurden. Im Fall von Sätzen als Einheit kommt ein regelbasierter heuristischer Ansatz namens Pragmatic Segmentizer [40] zum Einsatz, der sprachspezifisch arbeitet und für diesen Zweck in LERA eingebunden wurde. Der Segmentierungsprozess wird in Abschnitt 5.2 nochmals ausführlich betrachtet.

4.2.2 Normalisierung

Die Normalisierung der Textfassungen erfolgt je nach Aufgabe an mehreren Stellen im Arbeitsablauf von LERA und ist demnach kein eigenständiger Schritt wie in anderen Realisierungen des Gothenburg-Modells.
Um später Textvarianten rein aufgrund unterschiedlicher Kodierungen auszuschließen, werden beim Upload einer Datei ins System die folgenden zwei Normalisierungen vorgenommen. Zum einen wird die Kodierung des Inhalts automatisch in UTF-8 kodierten Unicode übertragen, beispielsweise wenn Textfassungen als ISO-8859-1 (Latin-1) oder als UTF-16LE kodierte Dateien in LERA eingespeist werden. Zum anderen bestehen gewisse Freiheitsgrade bei der Wahl der Unicodes zur Kodierung bestimmter Zeichen, die ebenfalls normalisiert werden müssen. Häufig entstehen solche Fälle im Zusammenhang mit Diakritika. Das kleine „ä“ kann beispielsweise als eigenständiger Unicode (U+00E4) oder als Kombination der Unicodes für den Buchstaben „a“ (U+0061) und für das (kombinierende) Trema „◌̈“

[3] Eine Auflistung der Elemente findet sich in LERAs Dokumentation unter `lera.uzi.uni-halle.de/wiki/import` (besucht am 18.08.2023).

(U+0308) kodiert werden. Gibt es mehrere Diakritika, ist zudem die Reihenfolge zu beachten. So kann beispielsweise ein ῷ, das heißt der griechische Kleinbuchstabe Omega mit einem Perispomene und einem Iota subscriptum, als einzelner Unicode „ῷ“ (U+1FF7), als die Abfolge „ω ◌͂ ◌ͅ“ (U+03C9 U+0342 U+0345) oder als die Abfolge „ω ◌ͅ ◌͂“ (U+03C9 U+0345 U+0342) kodiert werden. Während die verschiedenen Formen der Kodierung in ihrer Darstellung identisch sind, sind es beim maschinellen Textvergleich hingegen unterschiedliche Zeichen, sodass Editor:innen und Leserschaft der Ursprung daraus resultierender und in der Regel ungewollter Textvarianten oft nicht klar ist. Zur Behebung beider Probleme wird in LERA die *Normalization Form Canonical Decomposition* (NFD)[154] beim Upload einer Textfassung sicher gestellt, welche kombinierende Unicodes (wie Diakritika) als eigenständige Zeichen verwendet und zudem eine geordnete Reihenfolge festschreibt. Darauf aufbauend werden zudem Normalisierungen für die Suchfunktion von LERA vorgenommen, welche in 6.1 genauer beschrieben werden.

Die Wörter der importierten Textfassungen können über die grafische Nutzeroberfläche von LERA mit zusätzlichen Informationen angereichert werden. Mithilfe des ins System integrierten TreeTaggers [134] ist beispielsweise eine automatische und auf die jeweilige Sprache zugeschnittene Annotation der Wortgrundformen (Lemmata) und der lexikalischen Kategorien (Wortarten) möglich. Für den TreeTagger gibt es speziell trainierte Parameter für viele der von LERA unterstützten Sprachen (mit sehr unterschiedlicher Genauigkeit), welche für die Textfassungen entsprechend ihrer Sprache automatisch vom System gewählt werden. Die Parametersätze nutzen dabei in der Regel auch eigene Tagsets, das heißt eigene Listen definierter lexikalischer Kategorien, sodass eine weitere Normalisierung an dieser Stelle notwendig wird. In LERA werden die individuellen Elemente der verschiedenen Tagsets auf das *Universal Part-of-Speech Tagset* von Petrov et. al abgebildet, welches zwölf sprachübergreifende Kategorien definiert [113]. Dieses Vorgehen ermöglicht eine sprachunabhängige Nutzung der annotierten lexikalischen Kategorien.

Auch von den Editor:innen bereitgestellte Wörterbücher können für die Annotation der Textfassungen eingebunden werden. Beispiele hierfür sind das Auflösen von Abkürzungen oder das Auszeichnen von Eigennamen. Aber auch projektspezifische Annotationen, wie die Reduktion arabischer Verben auf ihre Wortwurzeln beim Projekt *AnonymClassic* (siehe Abschnitt 2.1.3), können mithilfe dieses Verfahrens realisiert werden. Die annotierten Zusatzinformationen können wiederum

zur Normalisierung beim Textvergleich oder bei den Visualisierungen zur Analyse der Textvarianten genutzt werden.

Eine weitere Normalisierung auf Basis von Wörterbüchern in LERA ist das Auflösen der Worttrennung am Zeilenende. Das Problem wurde bereits am Ende von Abschnitt 2.1.1 erwähnt und ist von folgender Gestalt: Ist ein Wort zu lang für eine Zeile, wird es an einem bestehenden Bindestrich im Wort oder mittels eines neu eingefügten Trennstrichs auf zwei Zeilen aufgeteilt. Die Information, ob ein Trenn- oder Bindestrich verwendet wurde, ist wichtig, wenn der Zeilenfall nachträglich aufgelöst werden soll, fehlt jedoch mitunter in den transkribierten Textfassungen. Die Auflösung dann heuristisch durchzuführen, beispielsweise unter der Annahme, es handle sich stets um Trennstriche, kann zu ungewollten Textvarianten führen, wie Abbildung 4.2 veranschaulicht.

… On lui accorda trois petits vaiſſeaux, & qua- tre-vingt-dix hommes. …	… On lui accorda trois petits vaiſſeaux, & quatre- vingt-dix hommes. …

Abbildung 4.2: Auszug aus der *Histoire des deux Index* mit unterschiedlichem Zeilenfall: Links wird ein Trennstrich eingefügt, rechts der bestehende Bindestrich genutzt, um die Worttrennung von „quatre-vingt-dix“ zu realisieren. Die Information, ob ein Trenn- oder Bindestrich vorliegt, ist wichtig, um beim Textvergleich ungewollte Varianten zu vermeiden.

Das gilt insbesondere in Sprachen wie Französisch, in denen Bindestriche häufig verwendet werden. Um das Problem zu lösen, wurde in LERA eine optionale Prozedur beim Import der Textfassungen integriert, die mithilfe eines Wörterbuchs eine exakte Auflösung vornehmen kann. Das benötigte Wörterbuch kann wiederum relativ zuverlässig für ein großes Textkorpus erzeugt werden, indem die Vorkommen der Wörter mit und ohne Bindestrich im Fließtext gezählt werden [121].

Für die erste Stufe des Textvergleichs – der Alignierung ähnlicher Textsegmente – werden die Textfassungen mithilfe der Token-Klassen auf ihren eigentlichen Inhalt reduziert, das heißt, überschüssige Leerzeichen, Strukturelemente und XML-Tags entfernt. Der Inhalt wird wiederum mit vorkonfigurierten Filtern normalisiert, was

die Umwandlung von Groß- in Kleinbuchstaben und das Entfernen von kombinierenden Zeichen, wie Diakritika, beinhaltet.

Für die zweite Stufe des Textvergleichs – dem detaillierten Vergleich der alignierten Textsegmente – können die Filter interaktiv über die grafischen Nutzeroberfläche ausgewählt werden.
Es gibt hierbei vier separate Kategorien für Buchstaben, Interpunktionen, diakritische Zeichen und andere Normalisierungen. Filter für Buchstaben sind oft einfache Ersetzungen, wie beispielsweise die Behandlung von Großbuchstaben (Apfel → apfel), Ligaturen (Stift → Stift) oder Vorkommen des langen s (ift → ist). Filter für Interpunktionen (Satzzeichen) oder Diakritika (zusätzliche Zeichen an Buchstaben) entfernen diese üblicherweise für den Vergleich. Die vierte Kategorie umfasst spezielle Transformationen, wie zum Beispiel die Abbildung von Wörtern auf ihre jeweilige, zuvor annotierte Grundform (liest → lesen) oder auch die in Abschnitt 2.1.1 motivierte Normalisierung von französischen Verben im Imparfait (vouloit → voulait).
In LERA gibt es darüber hinaus die Einteilung in Vergleichs- und Anzeigefilter. Während erstere in den Vergleich eingreifen und damit beeinflussen, was als Textvariante hervorgehoben wird, verändern letztere lediglich die Darstellung des Textes für die Nutzer:innen, um das Lesen zu vereinfachen. Viele Filter sind dabei sowohl als Vergleichs- als auch Anzeigefilter konzipiert, wie zum Beispiel die sprachabhängige Ersetzung vom kaufmännischen Und (&) mit der ausgeschriebenen Variante (& → et).
Die Filter in LERA arbeiten oft musterbasiert, das heißt, eine auf das Muster passende Zeichensequenz wird durch eine vorgegebene ersetzt. Zur vereinfachten Implementierung dieses Vorgehens kommen in den Mustern mitunter auch vordefinierte Zeichenkategorien des Unicode-Standards [155] zum Einsatz. Beispiele hierfür sind die Kategorien *Letter* für alle Arten von Buchstaben und *Mark*, welche alle Unicodes für Diakritika beinhaltet. LERAs Filter zum Entfernen aller Diakritika nutzt einen darauf basierenden regulären Ausdruck:

```
# finde einen einzelnen Buchstaben gefolgt von
# mindestens einem Diakritikum und behalte nur den Buchstaben
string.gsub!(/(\p{Letter})\p{Mark}+/, '\\1')
```

Da Filter oft sprachspezifisch sind, wird ihre Sichtbarkeit für Nutzer:innen automatisch von LERA in Abhängigkeit der Sprachen der zu vergleichenden Textfassungen gesteuert.

4.2.3 Alignierung von Textsegmenten

Nachdem die Textfassungen in den vorherigen Schritten einzeln betrachtet und aufbereitet wurden, beziehen sich die folgenden Schritte jeweils auf eine für die Kollationierung getroffene Auswahl mehrerer Fassungen.
Die Alignierung ist die erste von zwei Stufen, in die sich der Textvergleich in LERA unterteilt. Dabei werden die zuvor in der Segmentierung festgelegten Segmente der Textfassungen einander zugeordnet, wobei drei verschiedene Ansätze in LERA zur Verfügung stehen. Als Standardeinstellung kommt ein automatischer Ansatz zum Tragen, der die Zuordnung mittels einer Heuristik auf Basis der Ähnlichkeit der Segmente vornimmt. Dieser von André Medek (né Gießler) stammende Algorithmus wird im nachfolgenden Kapitel in Abschnitt 5.3.2 sowie ausführlich in [118] diskutiert. Zusätzlich zu dieser Standardeinstellung sind zwei weitere Alignierungsoptionen – via übereinstimmendem *xml:id*-Attribut und via gegebener Reihenfolge – in LERA verfügbar, die ebenfalls im nachfolgenden Kapitel näher beschrieben werden.

Eine der früh getroffenen, konzeptionellen Entscheidungen von LERA für die Kollationierung ist, dass Textfassungen als gleichrangig angesehen werden. Das bedeutet, dass es konzeptionell keine Leithandschrift oder Basistext beim Vergleich gibt und entsprechend kein Grund besteht, sich auf eine solche festlegen zu müssen. Dies gilt insbesondere auch für die Alignierung, bei der nach Übereinstimmungen zwischen den Segmenten aller Textfassungen gesucht wird, basierend auf ihrer Ähnlichkeit als dem wichtigsten Kriterium und nicht der Anordnung der Textfassungen.

Nachdem die Alignierung der Textsegmente durch LERA erfolgt ist, können Nutzer:innen interaktiv über die grafische Nutzeroberfläche eingreifen und den Vorschlag des Systems entsprechend der eigenen Auffassung anpassen. Ein typischer Anwendungsfall hierfür entsteht, wenn Autor:innen das Werk stark erweitern und beispielsweise aus einem Absatz in späteren Textfassungen mehrere werden (eventuell mit zusätzlichen Texteinschüben), sodass die automatische Alignierung auf Basis der vorgegebenen Segmentierung scheitern muss. Abbildung 4.3 zeigt dies schematisch an einem Beispiel.
Die manuellen Eingriffsmöglichkeiten umfassen das Auftrennen eines Textsegments in zwei oder mehr Segmente, das Verschmelzen von zwei oder mehr Segmenten und das lokale Ändern der Alignierung durch Verschiebung einzelner Segmente.

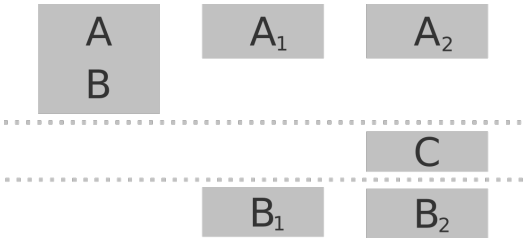


Abbildung 4.3: Das Textsegment B der ersten Textfassung kann nicht mit den ähnlichen Segmenten B_1 und B_2 aligniert werden, weil A und B in der ersten Fassung ein gemeinsames Segment bilden, das bereits aufgrund der angenommenen hohen Ähnlichkeit von A mit A_1 und A_2 aligniert wurde. In der dritten Textfassung wurde zudem mit C ein neues Segment dazwischen ergänzt. Für solche Fälle erlaubt LERA einen interaktiven Eingriff über die grafische Nutzeroberfläche.

Die durch die Editor:innen durchgeführten Eingriffe werden dabei in einem Protokoll aufgezeichnet und als ein aufklappbarer Dialog in LERAs Nutzeroberfläche aufgelistet, welcher zudem Schaltflächen für das Rückgängigmachen und Wiederherstellen anbietet. Darüber hinaus weisen optional zuschaltbare Symbole im Text auf die aufgetrennten beziehungsweise verschmolzenen Stellen hin. Die Möglichkeit dieses nachträglichen Eingriffs unterstreicht den semi-automatischen Ansatz, der mit LERA verfolgt wird.

4.2.4 Detailvergleich alignierter Segmente

Die zweite Stufe der Kollationierung ist der detaillierte Vergleich der in der ersten Stufe alignierten Textsegmente. In diesem Schritt findet die Berechnung der Textvarianten, das heißt der Unterschiede (und Gemeinsamkeiten) zwischen den Textfassungen, auf Wort- beziehungsweise Tokenebene statt. Die algorithmische Umsetzung wird im nachfolgenden Kapitel in Abschnitt 5.4 vorgestellt und diskutiert.
Bei der Ermittlung der Textvarianten gilt die gleiche konzeptionelle Entscheidung wie bei der Alignierung: Alle Textfassungen werden als gleichrangig betrachtet. Das heißt, zu jeder Variante zwischen zwei Fassungen werden stets auch die korrespondierenden Textpassagen in den anderen Fassungen bestimmt. Die so ermittelten Textvarianten – unter Berücksichtigung der ausgewählten Filter und Farbmodi – werden vom System gespeichert und dienen fortan als Datengrundlage für verschiedene Funktionalitäten von LERA. Dazu gehören die farbliche Hervorhebung der

Textvarianten in der synoptischen Gegenüberstellung der Textfassungen, die Auflistung der Varianten im kritischen Apparat, die Farbgebung in der Übersichtsleiste sowie die Generierung der verschiedenen Exportformate.

4.2.5 Analyse

LERA stellt für die Editor:innen sowie die spätere Leserschaft der Edition verschiedene Möglichkeiten zur Analyse der ermittelten Textvarianten bereit. Das umfasst zum einen die Visualisierungen in der grafischen Nutzeroberfläche LERAs, die sich in Close- und Distant-Reading-Ansätze unterteilen lassen. Hierbei bietet sich insbesondere die Möglichkeit des Zusammenspiels interaktiver Visualisierung zur explorativen Analyse der Textvarianten, wie sie in Kapitel 6 ausführlich diskutiert wird. Zum anderen ermöglicht LERA den Export der Vergleichsergebnisse in verschiedenen Formaten. Das umfasst direkte Darstellungen zur Präsentation, aber auch maschinenlesbare Formate, die wiederum als Datengrundlage für die systemexterne Analyse in anderer Software genutzt werden können.

Close-Reading

Close-Reading-Ansätze zur Darstellung erlauben einen Blick auf den tatsächlichen Volltext der verglichenen Textfassungen oder ihrer Varianten. Die grundlegende Visualisierung von LERA ist eine synoptische Gegenüberstellung der Textfassungen in Spaltenform, wobei diese segmentweise entsprechend der Ergebnisse der Alignierung unterteilt ist. Die Textvarianten wiederum, die beim Detailvergleich der alignierten Textsegmente gefunden wurden, werden durch farbliche Hervorhebungen angezeigt sowie zusätzlich in einem kritischen Apparat aufgelistet, der optional hinzu geschaltet werden kann. Abbildung 4.4 zeigt dies anhand eines Auszug der *Histoire des deux Indes*.
Der in LERA implementierte kritische Apparat enthält für jede Textvariante eine Liste mit den Siglen und den entsprechenden Passagen aller Textfassungen, wie ebenfalls in Abbildung 4.4 zu sehen ist. Diese listenförmige Darstellung führt das gleichrangige Vergleichsprinzip weiter, da im Kontrast zu den klassischen, kompakten Apparaten aus dem Buchdruck alle Textpassagen im Volltext wiederholt werden. Diese sehr lesefreundliche Form zieht dabei den Nutzen aus dem Platz, den digitale Medien bieten.

Abbildung 4.4: Auszug aus einer mittels LERA generierten Spaltensynopse für die *Histoire des deux Indes*. Die Textvarianten sind farblich (grau) hervor gehoben sowie im für den Absatz 225 aufgeklappten Variantenapparat gelistet.

Für die farblichen Hervorhebungen stehen sechs verschiedene Farbmodi zur Auswahl, die im Folgenden erläutert werden. Der Grundmodus hebt eine gefundene Textvariante samt der korrespondierenden Textpassagen in den anderen Fassungen grau hervor. Zudem werden die Hervorhebungen unabhängig vom Farbmodus mit fassungsübergreifenden Fußnoten-Indikatoren versehen, welche auf die entsprechende Stelle im kritischen Apparat verweisen. Abbildung 4.5 illustriert den Grundmodus anhand eines Beispiels aus der *Histoire des deux Indes*, wobei mittels des Filtersystems Varianten des langen/runden s, der Diakritika sowie Verbendungen im Imparfait ignoriert werden.

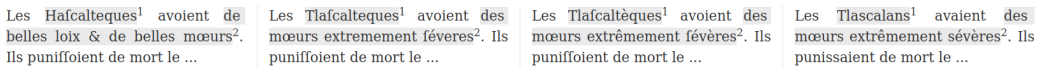


Abbildung 4.5: Der grundlegende Farbmodus von LERA markiert Textvarianten grau, wobei im Beispiel einige Varianten durch LERAs Filtersystem ausgeblendet werden.

Die übrigen Modi bauen auf dem Grundmodus auf und reichern die grau markierten Textvarianten mit zusätzlichen farblichen Hervorhebungen an. So können mit dem zweiten Modus die Editieroperationen aus den im Detailvergleich bereits berechneten Edit-Scripten angezeigt werden. Dabei werden zwischen aufeinander fol-

genden Textfassungen Einfügungen in Grün, Streichungen in Rot und Ersetzungen in Blau hervorgehoben, während kopierte Token auch weiterhin grau markiert werden. Diese zusätzliche Ebene zeigt die Operationen auf, mit der eine Textfassung in die darauffolgende überführt werden kann und ist somit abhängig von deren Reihenfolge, während durch die grauen Markierungen und den kritischen Apparat weiterhin der gleichrangige Vergleich der Textfassungen visualisiert wird. Abbildung 4.6 veranschaulicht diesen Farbmodus am Beispiel.

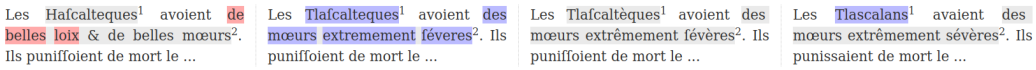


Abbildung 4.6: Die Editieroperationen zwischen aufeinander folgenden Textfassungen können farblich hervorgehoben werden. Im Beispiel müssen die rot markierten Token gelöscht und die blau markierten ersetzt werden, um die erste in die zweite Textfassung zu überführen.

Ein dritter Farbmodus wird in Abbildung 4.7 veranschaulicht. Dieser hebt ausschließlich Token hervor, die in exakt einer Textfassung vorkommen, was auf eine gewisse Unabhängigkeit der Fassung bezogen auf die anderen hindeutet. Solche Einzelvorkommen können wichtige Evidenz zum Nachvollziehen der Filiation und Textgenese liefern.

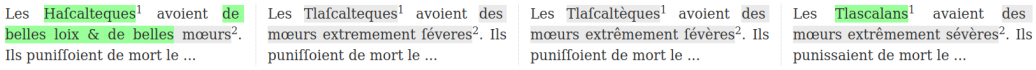


Abbildung 4.7: Einzelvorkommen können mit dem entsprechenden Farbmodus grün hervorgehoben werden.

Mit dem in Abbildung 4.8 illustrierten, vierten Farbmodus können analog jene Token hervorgehoben werden, welche in exakt zwei Textfassungen vorkommen. Dabei wird automatisch eine Farbe für jedes Paar von Textfassungen generiert, um sie unterscheidbar zu halten. Das Aufzeigen solcher exklusiven Paare kann wiederum wichtige Indizien dafür liefern, dass zwei Textfassungen eng miteinander verwandt sind.
Der fünfte Modus weitet dieses Prinzip auf Gruppen übereinstimmender Token aus, indem diese in blauen Farbtönen mit unterschiedlicher Sättigung eingefärbt werden.

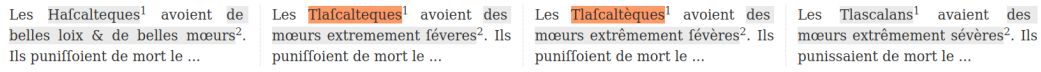


Abbildung 4.8: Im Beispiel wurde das Wort „Tlafcalteques“ durch einen speziellen Farbmodus zum Anzeigen von Paaren hervorgehoben, da es (nach Normalisierung der Akzente) in exakt zwei Textfassungen vorkommt.

In diesem Gruppen-Modus gilt dabei, je seltener der Token in den Textfassungen ist, desto kräftiger der Farbton. Abbildung 4.9 zeigt dies am Beispiel.

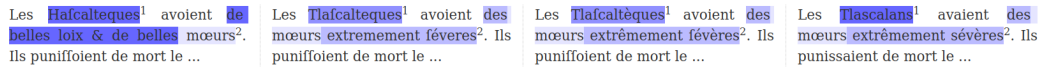


Abbildung 4.9: Mittels blauer Farbtöne wird der Grad an Übereinstimmung zwischen den Textfassungen verdeutlicht.

Ein zusätzlicher, sechster Modus verzichtet gänzlich auf farbliche Hervorhebungen und zeigt nur die Indikatoren zur Referenzierung auf den Variantenapparat an.
Als Alternative zur spaltensynoptischen wurde eine zeilensynoptische Darstellung in LERA integriert. Diese auch als Partitursynopse bekannte Darstellungsform des Textvergleichs ist vor allem für Editionen mit sehr vielen Textfassungen, aber kurzen Textsegmenten geeignet. Im Rahmen dieser Darstellung findet auch eine Alignierung der identischen beziehungsweise der variierenden Passagen innerhalb der Textsegmente statt, sodass die Unterschiede zusätzlich hervorstechen. Praktikable Segmentgrößen für diese Darstellungsform sind Sätze oder kurze Absätze. Bei einer Überschreitung der verfügbaren Breite des Bildschirms ermöglicht eine horizontale Scrollleiste die Navigation innerhalb des Segments, wobei die Siglen der Textfassungen ihre Position beim Scrollen beibehalten. Auch in der Partitursynopse sind Filter, Farbmodi und weitere Einstellungen analog zur Spaltensynopse verfügbar. Abbildung 4.10 zeigt die Partitursynopse für ein Segment aus *Keter Shem Ṭov*.

Distant-Reading

Neben den beschriebenen Darstellungen der Textfassungen und -varianten im Volltext verfügt LERA über zwei optional zuschaltbare Distant-Reading-Visualisierungen, die hier zunächst kurz vorgestellt und im Kapitel 6 im Detail diskutiert wer-

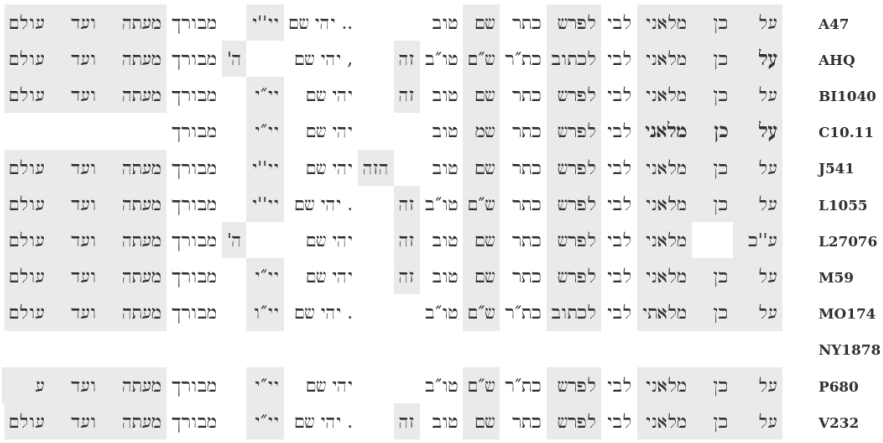


Abbildung 4.10: Auszug einer mittels LERA erstellten Partitursynopse eines Segments von *Keter Shem Ṭov* in zwölf Textfassungen. Die Schreibrichtung der Texte ist von Rechts-nach-Links, sodass die Siglen der Textfassungen entsprechend der Lesegewohnheiten automatisch vom System rechts platziert wurden.

den. Beide Visualisierungen sind interaktiv und nutzen den jeweils aktuellen Berechnungsstand der Alignierung des Detailvergleichs.

Die erste ist eine integrierte Übersichts- und Navigationsleiste namens CATview [117, 119], welche ausführlich in Abschnitt 6.2 diskutiert wird. CATview gibt einen Überblick über die Spalten- beziehungsweise Partitursynopse, indem die Textsegmente der verglichenen Textfassungen als Rechtecke entsprechend der aktuellen Alignierung angeordnet dargestellt werden. Dabei werden verschiedene Farbintensitäten zur Veranschaulichung der Textvarianten genutzt: Je dunkler die Farbe eines Rechtecks ist, desto variantenreicher ist das dazugehörige Textsegment in Bezug auf die alignierten Segmente der anderen Fassungen. Durch Anklicken eines Rechtecks scrollt die synoptische Darstellung zu dem ausgewählten Segment. Zusätzlich wird die aktuelle Scroll-Position durch eine Markierung innerhalb von CATview angezeigt. Abbildung 4.11 zeigt CATview für Livre VI der *Histoire des deux Indes* in den vier Textfassungen.

Die zweite Distant-Reading-Visualisierung in LERA sind interaktive Wortwolken, wobei jeweils eine pro Textfassung generiert und angezeigt wird (siehe Abschnitt 6.3). Wortwolken stellen die vermeintlich relevantesten Wörter in Bezug auf eine Reihe gewählter Optionen dar. Zum Beispiel können die relevantesten Wörtern auf bestimmte Wortarten beschränkt oder das Relevanzmaß selbst angepasst werden.

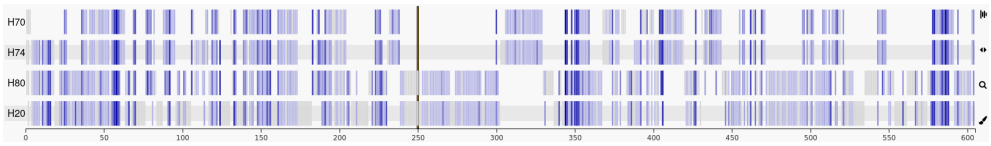


Abbildung 4.11: CATview für das gesamte sechste Buch der *Histoire des deux Indes*.

Die Wortwolken, CATview sowie eine ebenfalls in LERA integrierte Suchfunktion wurden zur interaktiven Analyse der Textvarianten miteinander verknüpft. Die durch dieses Zusammenspiel ermöglichte explorative Form der Analyse sowie die aktuellen Funktionalitäten und Kombinationsmöglichkeiten der drei Komponenten werden in Kapitel 6 genauer vorgestellt und diskutiert. Zudem greift ein Konferenzbeitrag von 2016 dieses Thema auf [139].

Darstellung transponierter Segmente

Eine Sonderrolle nimmt die Darstellung von transponierten Textsegmenten ein, die in LERA als eine Mischung aus Close- und Distant-Reading realisiert wurde. Unter transponierten Segmenten werden hier Textsegmente verstanden, die zwar ähnlich zueinander sind, aber nicht aligniert wurden beziehungsweise werden konnten. Eine mögliche Ursache – die Segmentierung wurde während der Überarbeitung eines Werks verändert, sodass die Alignierung mit den vorgegebenen Segmentgrenzen nicht funktionieren kann – wurde bereits in Abschnitt 4.2.3 beschrieben und mittels manueller Eingriffsmöglichkeiten in die Alignierung behandelt. Ein damit verwandtes, aber grundlegenderes Problem für die Kollationierung sind echte Transpositionen, das heißt Vertauschungen der Segmente. Sie finden sich häufig bei stark überarbeiteten Textfassungen, bei denen größere Abschnitte umgestellt wurden. Weicht die Reihenfolge der Segmente zwischen den Textfassungen ab oder erscheinen ähnliche Segmente in einem gewissen Abstand zueinander, so kann ihre Alignierung durch andere, bereits zuvor alignierte Segmente blockiert werden. Folglich können sie nicht direkt nebeneinander in der Synopse platziert werden, ohne die Chronologie einer Textfassung aufzubrechen. Abbildung 4.12 veranschaulicht das Problem schematisch an einem kurzen Beispiel.
In LERA wird das Phänomen der transponierten Segmente an mehreren Stellen im System behandelt. Die Erkennung findet bereits bei der Alignierung statt, indem der Algorithmus bei der Zuordnung der Segmente mittels Heuristik oder xml-id-

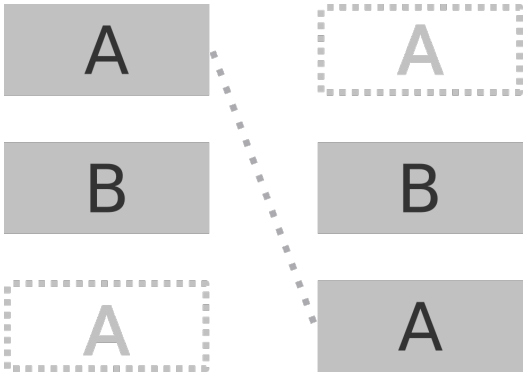


Abbildung 4.12: Das Problem von transponierten Textsegmenten schematisch dargestellt. Durch die Alignierung eines Segments (B) kann das andere (A) nicht mehr aligniert werden, ohne die Reihenfolge der Segmente in den Textfassungen zu verändern. In LERA wird diesem Problem durch das Anzeigen von Kopien der betroffenen Segmente begegnet (gestrichelt dargestellt).

Attribut die ähnlichen aber blockierten Zuordnungen aufzeichnet und speichert. Auf dieser Basis werden nun Kopien der transponierten Segmente an der jeweiligen Stelle platziert, wenn diese nicht bereits durch ein real zugeordnetes Segment belegt ist. Die Kopien werden gesondert dargestellt. In Zeilen- sowie Partitursynopse, im kritischen Apparat sowie einigen Exportformaten werden sie beispielsweise statt in der üblichen schwarzen Schriftfarbe in einem hellen Grau angezeigt. Innerhalb der Synopse in LERAs Nutzeroberfläche wird zudem eine Verlinkung auf ihre tatsächliche Position beigefügt. Zudem werden die Kopien in CATview als ungefüllte Rechtecke leicht angedeutet und visuell mit ihrer eigentlichen Position verknüpft. Abbildung 4.13 zeigt die Darstellung transponierter Segmente als Kopien in LERA.

Die Darstellung der Kopien und ihre Einbeziehung in den Textvergleich ist optional und kann jederzeit über die Nutzeroberfläche an- und abgeschaltet werden. LERA bietet zudem die Möglichkeit, die Platzierung transponierter Segmente als Kopien erneut zu berechnen. Dies wird notwendig, wenn sich die Datengrundlage durch den manuellen Eingriff in die Alignierung geändert hat.

Abbildung 4.13: Aufgrund einer Transposition konnte dem Segment 464 das Segment 525 aus den späteren Textfassungen nicht direkt zugeordnet werden. LERA bietet für diese Fälle die Option an, Kopien der entsprechenden Segmente anzuzeigen, wie dies am Beispiel sowohl im Volltext als auch in der Übersichtsleiste zu sehen ist.

4.3 Schnittstellen

Dieser Abschnitt diskutiert die in LERA integrierten Schnittstellen für den Im- und Export verschiedener Daten. Dabei unterstützt LERA nach langjähriger Entwicklung eine große Bandbreite, die sowohl allgemein verbreitete Austauschformate als auch projektspezifische Lösungen umfasst. Der folgende Abschnitt ist dreigeteilt und beschreibt den Import und den Export von Daten über der grafischen Nutzeroberfläche sowie als dritten Zugang die verfügbaren Programmierschnittstellen.

4.3.1 Import via Nutzeroberfläche

Wesentlich für die Arbeit mit LERA sind die Textfassungen eines Werks, die entsprechend zunächst in das System importiert werden müssen. Wie bereits im Abschnitt zur Tokenisierung (4.2.1) und zur Normalisierung (4.2.2) aufgeführt, unterstützt LERA dabei einfache Textdateien (TXT) sowie nach den Richtlinien der TEI codierte XML-Dateien, die über die grafische Nutzeroberfläche importiert werden können.

Als zweite Datengrundlage lassen sich Wörterbücher in LERA importieren, die zur

Normalisierung der Textfassungen beim Textvergleich genutzt werden können. Dafür wurde ein spezielles JSON-Format spezifiziert, das neben den eigentlichen Einträgen auch Angaben zum Typ des Wörterbuchs und zur notwendigen Normalisierung – beispielsweise die konsequente Kleinschreibung der Einträge – bei dessen Anwendung enthält.

LERA ermöglicht darüber hinaus den Import von bereits durchgeführten Textvergleichen über die grafische Nutzeroberfläche. Das ist nützlich, wenn Teilergebnisse mittels externer Software berechnet wurden und zur Visualisierung oder Weiterverwendung in das System zurückgespielt werden sollen. Dazu steht zum einen ein einfaches JSON-Format zur Verfügung, welches neben einigen Metadaten die Kodierung einer Segment-Alignierung ermöglicht. Als Zweites wurde ein komplexeres Austauschformat für in LERA erstellte Editionen beziehungsweise Synopsen definiert, das die verschiedenen, benötigten Einzeldateien als ZIP-Archiv komprimiert zusammenfasst. Dazu gehören zunächst die Textfassungen, die als XML-Dateien abgespeichert werden. Waren diese zuvor einfache Textdateien, wird beim Export ein rudimentäres Grundgerüst nach TEI-Konvention ergänzt. Lagen sie hingegen bereits in XML vor, werden die ursprünglich hochgeladenen Dokumente als Grundlage genutzt. In die Textfassungen werden die aktuellen Segmentier- und Alignierungsinformationen mithilfe zusätzlicher Elemente eingefügt. Hinzu kommt eine JSON-Datei, welche zusätzliche Angaben zur Edition enthält, wie Metadaten zu den Textfassungen und ihrer Anordnung innerhalb der Edition, Metadaten zur Edition selbst und die Parameter zum Textvergleich, die LERA wiederum beim Import benötigt.

4.3.2 Export via Nutzeroberfläche

Für den Export von Editionen, das heißt den synoptischen Vergleich mehrerer Textfassungen, stehen verschiedene Formate in der Nutzeroberfläche zur Verfügung. Bei einigen Formaten kann ausgewählt werden, ob die gesamte Synopse oder nur einzelne Segmente exportiert werden sollen. Zum einen ermöglicht LERA den Export zweier unterschiedlicher PDF-Formate. Eins ahmt die spaltensynoptische Darstellung von LERA nach. Entsprechend der zuletzt gewählten Filtern werden die Textvarianten dabei wie im Vorbild farblich hervorgehoben und in einem kritischen Apparat aufgelistet. Dieses Exportformat liefert eine Art Protoedition, die als Grundlage für den späteren Druck dienen kann. Das zweite PDF-Format ist eine kompaktere Darstellung, welche die Alignierung ähnlicher Textsegmente als Volltext, zusam-

men mit einer fortlaufenden Nummerierung, wiedergibt, wobei mehrere Segmente pro Seite platziert werden. Diese kompaktere Form ist für den sparsamen Druck von Zwischenständen ausgelegt, den Editor:innen so auf Papier annotieren und bearbeiten können. Beide Formate werden vom System mittels XeLaTeX generiert. Der generierte Quellcode kann dabei ebenfalls über die Nutzeroberfläche heruntergeladen werden.

Als weitere Exportmöglichkeit sind verschiedene Ausprägungen von HTML verfügbar, welche entweder für eine vollständige Synopse oder ein ausgewähltes Textsegment generiert werden können. Dabei wird der aktuelle Bearbeitungsstand aus LERA entsprechend der gewählten Vergleichsparameter extrahiert. Das geschieht entweder in Form einer einzelnen HTML-Datei[4] oder in Form eines ZIP-Archivs, welches optional auch die interaktiven Visualisierungen von LERA enthält[5]. Hinzu kommen Optionen, um beispielsweise Segmentnamen und kopierte Segmente ein- oder auszublenden sowie um die Synopse und den Apparat von Rechts-nach-Links anzuzeigen. Der HTML-Export bietet sich als Teil digitaler Editionen an, die im Web präsentiert werden.

Als dritte Gruppe wurden zwei exportierbare XML-Formate in LERA integriert, die wiederum direkt als Importformat für die beiden, in den digitalen Editionswissenschaften verbreiteten Werkzeuge TEI Publisher [41] und Versioning Machine [138] genutzt werden können. Beide orientieren sich dabei an den Richtlinien der Text Encoding Initiative (TEI) und sind auch unabhängig der avisierten Programme nutzbar. Die Integration der XML-Formate erhöht die Interoperabilität von LERA. Neben XML stehen mit den in Abschnitt 4.3.1 bereits beschriebenen JSON- und ZIP-Formaten zwei weitere maschinenlesbare Formate für die externe Weiterverarbeitung bereit, die über LERAs Nutzeroberfläche exportiert werden können. Beide

[4] Diese Form wurde beispielsweise von Dr. Andrejka Žejn in ihrer als digitale Plattform publizierten Studie zur Entwicklung der slowenischen Erzählungen *Izhodišča slovenske pripovedne proze* [157] genutzt, welche auch einen Vergleich sechs unterschiedlicher Textfassungen der von Janez Cigler (* 1792 in Vodmat bei Ljubljana, † 1867 in Višnja Gora) verfassten Kurzgeschichte *Sreča v nesreči* (*Das Glück im Unglück*) enthält. So wurden einige mittels LERA erstellte Synopsen direkt als verlinkte Unterseiten in die Plattform integriert.

[5] Ein Beispiel hierfür ist die Musteredition einiger arabischer Manuskripte von *Kalīla and Dimna*, die unter `https://kd-preview-edition.geschkult.fu-berlin.de/lera` (abgerufen am 11.08.2023) verfügbar ist.

enthalten Angaben zur mittels des Textvergleichs generierten Segment-Alignierung und können wiederum in LERA importiert werden.

4.3.3 Programmierschnittstellen

Neben den Im- und Exportmöglichkeiten über die grafische Nutzeroberfläche besitzt LERA noch einige programmierbare Schnittstellen, die einen automatisierten Datenaustausch ermöglichen.

Dazu wurde zum einen die Möglichkeit geschaffen, mittels der HTTP-Methoden POST und GET Textfassungen in Form von JSON-Objekten in das System zu importieren beziehungsweise daraus zu exportieren.
Mit dieser Schnittstelle wurde beispielsweise eine externe Software aus dem Projekt *AnonymClassic* angebunden, welche die Verwaltung und Bearbeitung der Textfassungen übernimmt. Über den Textvergleich in LERA fallen leicht Transkriptionsfehler auf, die in den Textfassungen korrigiert werden sollen. LERA kann dazu so konfiguriert werden, dass die Textsegmente in der synoptischen Gegenüberstellung direkt mit dem Bearbeitungsmodus in der externen Software verlinkt sind. Nach der Korrektur wird wiederum die JSON-API genutzt, um die aktualisierte Textfassung automatisch zurück in LERA zu importieren und den Datenbestand so zu synchronisieren.

Zum anderen wurde im Rahmen der Edition von *Keter Shem Ṭov* ein konfigurierbares Modul in LERA integriert, welches die Synchronisation des Datenbestandes mit einem externen Repositorium ermöglicht. Dieses prüft auf Knopfdruck das Vorhandensein neuer sowie die Änderung bestehender Textfassungen im Repositorium und importiert die Änderungen in LERAs Datenbestand. Die technische Umsetzung wird in [115] genauer beschrieben.

In beiden Fällen ist eine automatische Aktualisierung der Daten in LERA notwendig, was sich sowohl auf die Textfassungen als auch auf bestehende Textvergleiche, welche diese enthalten, bezieht. Dazu werden die Token der aktualisierten Textfassungen mithilfe der ohnehin in LERA verfügbaren Algorithmen zum Textvergleich jener der bestehenden Versionen zugeordnet, sodass eine punktuelle Aktualisierung vorgenommen werden kann.

4.4 Software-Architektur

LERA wurde als Webanwendung konzipiert und mittels des Ruby-on-Rails [61] Frameworks realisiert. Die webbasierte Realisierung – eine Server-Client-Architektur – spiegelt sich dabei in einer Zweiteilung der Software in Backend und Frontend wider, wobei diese auf verschiedenen Maschinen laufen können.
Ganz konkret sehen Nutzer:innen von LERA nur das Frontend als im Browser gerenderte HTML-Dateien, welche die grafische Nutzeroberfläche formen und von interaktiven Elementen durchsetzt sind. Dazu gehören beispielsweise Schaltflächen zum Dokumentupload und deren Verwaltung, die Auswahl der Textfassungen zum Generieren einer vergleichenden Synopse, die Analyse dieses Textvergleichs mit interaktiven Visualisierungen und schließlich der Download der Ergebnisse in verschiedenen Formaten. Die Interaktivität des Frontends wird mittels Javascript, erweitert um einige Bibliotheken wie jQuery [47] und d3.js [15], realisiert.
Das Frontend sendet wiederum Anfragen an das Backend, auf dem die aufwändigen Berechnungen und Algorithmen zum Textvergleich laufen. Darüber hinaus verwaltet es die Daten mithilfe einer MySQL-Datenbank [30]. Grundlegend sind das die in Token zerlegten Textfassungen sowie die darauf aufbauenden Synopsen. Die Datenbank erlaubt das persistente Speichern der anfallenden Daten und einen effizienten, punktuellen Zugriff drauf entsprechend der Nutzeranfragen.
Der webbasierte Ansatz hat durch die Auftrennung in Back- und Frontend einige hilfreiche Eigenschaften, die der Umsetzung der beschriebenen Zielsetzung für LERA dienlich sind. Ein wesentlicher Aspekt betrifft die Installation und damit die Nutzerfreundlichkeit der Software. So kann die technisch anspruchsvolle Einrichtung des Backends vor den eigentlichen Nutzer:innen ferngehalten werden. Diese können hingegen mit einem gängigen Browser auf LERAs grafische Nutzerüberfläche ohne eigene Installation der Software zugreifen. Das ist gleichzeitig auch ein Vorteil gegenüber anderer Software dieser Domäne. Zur Realisierung wird LERA auf den Servern der Uni Halle gehostet und interessierten Editionsprojekten als Software-as-a-Service bereitgestellt. Daraus ergibt sich ein weiterer Vorteil aus Sicht der Nutzer:innen: So wird keine eigene leistungsstarke Hardware benötigt, da die aufwändigen Berechnung und die Datenverwaltung auf dem Server stattfindet. Zudem erlaubt die persistente Speicherung der Daten auf dem Server ein verteiltes Arbeiten mit LERA von verschiedenen Clients aus.
Um LERA als Echtzeitanwendung zu realisieren, wurden verschiedene Techniken zur Verbesserung der Reaktionszeiten eingesetzt. Im Frontend gehört dazu beispiels-

weise das partielle Rendern von HTML-Seiten. Diese Technik ist sinnvoll, um Nutzer:innen während aufwändiger Berechnungen bereits Teilergebnisse zur Verfügung zu stellen oder nur einzelne Fragmente größerer HTML-Seiten auszutauschen, statt ein komplettes Neurendern durchzuführen. Im Backend zählt dazu neben dem Einsatz effizienter Algorithmen (siehe Kapitel 5) insbesondere die Modellierung und der Zugriff auf die Datenbank[6].

4.4.1 Konfigurationsmöglichkeiten

Mit den unterschiedlichen Editionsprojekten gehen auch verschiedene Einsatzszenarien von LERA und damit der Wunsch nach einem anpassbaren Funktionsumfang der Software einher. Um diesen zu erfüllen, wurde LERA durch eine optionale Konfigurationsdatei im YAML-Format umfangreich anpassbar gestaltet. Die um die 140 Einzeloptionen der Konfigurationsdatei ermöglichen dabei verschiedene Arten der Anpassung.

Zum Ersten können einzelne Funktionalitäten an- und abgeschaltet werden, beispielsweise das Neuanlegen, Modifizieren oder Entfernen von Dokumenten, Wörterbüchern oder Editionen. Zu letzterem zählen insbesondere auch die erlaubten Optionen für den Textvergleich. So kann zum Beispiel der Vergleich auf Tokenebene erlaubt werden, während eine feste Alignierung vorgegeben wird. Auch welche Visualiserungsformen verfügbar sind, lässt sich so steuern.

Als Zweites können über die Konfigurationsdatei Standardwerte festgelegt werden. Dazu gehört, welche Einheit initial für die Segmentierung gewählt wird und welche Filter und welcher Farbmodi standardmäßig beim Textvergleich gesetzt sind.

Zum Dritten lässt sich die Nutzeroberfläche bis zu einem gewissen Grad modifizieren, beispielsweise ob eine beschreibende Startseite zu LERA, eine Sprachauswahl für die Oberfläche (deutsch und englisch) oder weiterführende Informationen in Form eines Wikis verfügbar gemacht werden. Zudem lässt sich eine abweichende Schriftart für die Darstellung der Textfassungen festlegen sowie mittels einer zusätzlichen Lokalisierungsdatei eine Umbenennung der Steuerelemente und Bezeichnungen vornehmen.

Als Viertes kann auch die Datenverarbeitung bis zu einem gewissen Grad durch die Konfigurationsdatei gesteuert werden. Dazu gehört insbesondere der Tokenizer mit

[6] Beispielsweise durch die Verwendung komplexer Indizes und Bulk-Operationen sowie dem direkten Arbeiten auf den Daten statt der Erstellung Ruby-on-Rails typischer Active Record Objekte.

Optionen zum Ignorieren von XML-Tags oder dem Definieren einer Vorverarbeitung hochgeladener XML-Dateien via XSLT.
Des Weiteren gibt es einen integrierten Synchronisationsmechanismus in LERA, der den Datenbestand mit einem externen Werkzeug oder Repositorium abgleichen kann und mittels der Konfigurationsdatei spezifiziert wird.
Zuletzt kann als eine wesentliche Festlegung auch die Art der Datenhaltung konfiguriert werden. Dabei gibt es drei Optionen. Standardmäßig sind in LERA die Daten für alle Nutzer:innen einer Instanz über eine gemeinsame Datenbank verfügbar. Diese gemeinsame Datenhaltung ist für den kollaborativen Einsatz der Software in Editionsprojekten mit kleinen Teams gedacht. Im Kontrast dazu lässt sich auch eine separate Datenhaltung wählen, bei der Nutzer:innen nur mit eigenen Daten arbeiten können. Die Zuordnung erfolgt dabei im Moment allerdings via Cookie, sodass den Nutzer:innen ein spezifischer Browser zugeordnet wird. Neben den eigenen Daten ist es auch möglich, Dokumente und Editionen vorzugeben. Abhängig von freigeschalteten Funktionalitäten LERAs arbeiten die Nutzer:innen in der Regel auf vollen Kopien der Daten, sodass ein komplettes Umarbeiten der vorgegebenen Beispiele möglich ist. Die Konfiguration eignet sich für den Einsatz der Software in Veranstaltungen mit einer abschätzbaren Anzahl von Teilnehmer:innen, wie der Lehre, Workshops oder Sommerschulen. Alternativ kann LERA aber auch so konfiguriert werden, dass nur die benötigten Teildaten für einen reduzierten Funktionsumfang kopiert werden. Das ist sinnvoll für Online-Demos, bei der es zu sehr vielen Anfragen kommen kann. Beispiele hierfür sind die Demo von LERA selbst [120] oder der Kritischen Gesamtausgabe der Werke Hannah Arendts [58]. Bei beiden ist eine fixe Segmentierung und Alignierung vorgegeben, aber eine Anpassung vom Detailvergleich möglich (Filter / Farbmodi). Um die Datenhaltung noch effizienter zu gestalten, werden Kopien nur dann angelegt, wenn durch die jeweiligen Nutzer:innen tatsächlich etwas angepasst wird.

Kapitel 5

Berechnung von Textvarianten

Die Algorithmen zur Kollationierung der Textfassungen eines Werkes sind zentrale Komponenten von LERA. Dieses Kapitel beschreibt einleitend den gewählten Ansatz, die Kollationierung in mehrere Stufen einzuteilen und darauf aufbauend die algorithmische Umsetzung der drei wesentlichen Schritte. Nur geringen Platz nimmt hierbei der erste Schritt – die Segmentierung der Textfassungen – ein. Den beiden eigentlichen Vergleichsstufen – der Alignierung und der detaillierten Differenzanalyse – wurde mit durch im Projekt *Semi-automatische Differenzanalyse von komplexen Textvarianten (SaDA)* neu entwickelten Algorithmen für den Einsatz in LERA begegnet. Dabei wird in diesem Kapitel der wesentliche Fokus auf den Algorithmus zur detaillierten Differenzanalyse gelegt. Dazu gehört, neben der theoretischen Betrachtung des zu lösenden Kernproblems, die ausführliche Vorstellung der dazu neu entwickelten Heuristik samt der für den praktischen Einsatz in LERA implementierten Erweiterungen.

5.1 Mehrstufige Kollationierung

Die Berechnung von Textvarianten in LERA lässt sich als eine Abfolge von drei Schritten gliedern, die über die Nutzeroberfläche gesteuert werden können. Der erste Schritt ist eine Einteilung der zu vergleichenden Textfassungen in größere Textsegmente, beispielsweise Sätze oder Absätze. Im zweiten Schritt, beziehungsweise der ersten Vergleichsstufe, werden diese Textsegmente einander zugeordnet. Diesem Vergleich auf Makroebene folgt mit der detaillierten Differenzanalyse schließlich der Vergleich auf Mikroebene. In diesem dritten Schritt werden die Textvari-

anten zwischen den alignierten Segmenten auf Wort- beziehungsweise Tokenebene bestimmt. Abbildung 5.1 illustriert den hierarchischen Ansatz von LERA bei der Kollationierung.

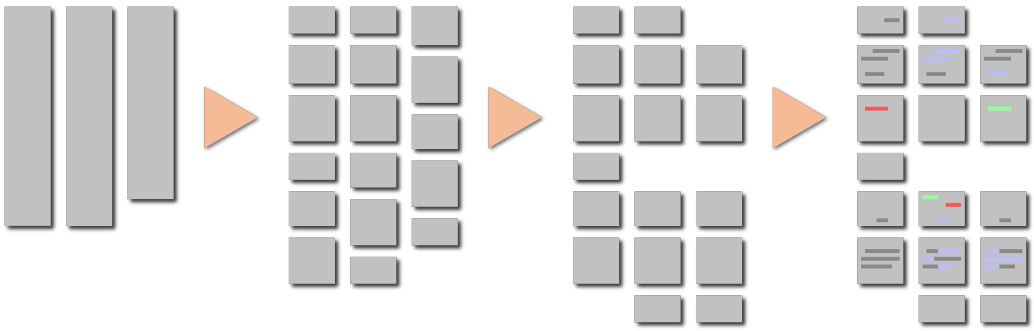


Abbildung 5.1: Schematische Darstellung von LERAs Ansatz zur mehrstufigen Kollationierung. Zunächst werden die Textfassungen in Segmente zerlegt, die in einer ersten Vergleichsstufe aligniert werden, bevor schließlich in der detaillierten Differenzanalyse als der zweiten Vergleichsstufe die Textvarianten auf Wort- beziehungsweise Tokenebene bestimmt und dargestellt werden.

Wie in Abschnitt 4.2 bereits dargelegt, erweitert das in LERA verfolgte, hierarchische Vorgehen bei der Kollationierung den entsprechenden Schritt der Alignierung aus dem Gothenburg-Modell.
Die Aufteilung in eine Alignierung größerer Textsegmente und die anschließende detaillierte Differenzanalyse dieser einander zugeordneten Textsegmente hat dabei den wesentlichen Vorteil, dass die Verarbeitung und Darstellung umfangreicher Textfassungen erleichtert wird. So bleiben umfangreiche Werke zum einen für Editor:innen und die Leserschaft handhabbar, da die Textsegmente in der Regel klein genug sind, um beim Lesen inhaltlich erfasst werden zu können und gleichzeitig lang genug, um genügend Kontext zur Einordnung der darin vorkommenden Textvarianten zu liefern. Zum anderen hat dieser Teile-und-Herrsche-Ansatz positive Auswirkungen beim algorithmischen Bestimmen der Textvarianten. So ist die Anwendung aufwändiger Verfahren, wie beispielsweise das Bestimmen der Levenshtein-Distanz, nur auf kleinen Textmengen praktikabel. Durch das hierarchische Vorgehen können sie im Rahmen der zweiten Vergleichsstufe auch bei umfangreichen Werken angewendet werden.

5.2 Segmentierung

Zur Kollationierung in LERA werden die Textfassungen in einem ersten, vorangestellten Schritt in Segmente zerlegt. Arbeitsgrundlage hierfür ist wiederum die initiale Zerlegung der Textfassungen in Token, wie in Abschnitt 4.2.1 beschrieben, welche beim Upload der Dateien ins System automatisch durchgeführt wird. Die dabei erkannten Tokenklassen inklusive struktureller Bestandteile sowie der Verknüpfung von öffnenden und schließenden XML-Tags bei hochgeladenen XML-Dateien erlauben schließlich eine darauf aufbauende Einteilung der Textfassung in Segmente, indem die Token entsprechend einer von den Nutzer:innen wählbaren Segmentieroption wieder zu Segmenten gruppiert werden.
Diese Zweiteilung in Tokenisierung und Segmentierung ist nützlich, um Nutzer:innen die Möglichkeit zu geben, Segmentgrenzen neu zu berechnen beziehungsweise manuell anpassen zu lassen, ohne die Dokumente erneut in Gänze verarbeiten zu müssen.

Prinzipiell wird eine Textfassung als Folge von Token betrachtet. Die Segmentierung ermittelt darin zunächst jene Token, die zur Grenzziehung genutzt werden. Das geschieht in Abhängigkeit der gewählten Segmentieroption in der Regel anhand der Informationen, die bei der Tokenisierung gesammelt wurden, wie der Tokenklasse oder dem textuellen Inhalt eines Tokens.

Drei typische Beispiele sind die Segmentierung in Zeilen, Sätze oder Absätze. Zeilenumbrüche werden direkt bei der Tokenisierung erkannt und als eigene Tokenklasse gespeichert, sodass die entsprechende Segmentieroption nur die Tokenklasse abfragen muss. Satzgrenzen sind hingegen komplexer und zudem abhängig von der Sprache. Für diese Segmentieroption wurde ein bestehender, heuristischer Ansatz namens Pragmatic Segmentizer [40] in LERA eingebunden, welcher neben dem Inhalt der Token auch die Sprache der Textfassung erhält und dann mithilfe sprachspezifischer Regeln den gegebenen Text in Sätze zerlegt. Beim dritten Beispiel (Absätze) kann wieder auf die Tokenklasse zurückgegriffen werden, da sie bei der Tokenisierung erkannt werden. Es gibt aber für XML-Dateien auch die alternative Segmentieroption anhand von <p>-Tags, die jeweils zusammen mit ihren schließenden Tags, welche bei der Tokenisierung bereits erkannt wurden und via Datenbankabfrage direkt zugänglich sind, einen Bereich definieren.
An diesen drei Beispielen wird deutlich, dass eine gefundene Grenze in Abhängig-

keit von der gewählten Option vom Segmentierer unterschiedlich zu interpretieren ist, nämlich ob es sich dabei um einen Endpunkt (Zeilen, Sätze) oder Beginn eines Bereichs (<p>-Tag) handelt. Zudem gibt es Startpunkte, wie beispielsweise bei der Segmentierung an <milestone>-Tags.

Die geeignete Segmentieroption hängt vom Inhalt und Genre der Textfassungen sowie ihrer Transkriptionen und den untersuchten Forschungsfragen ab, welche sich je nach Editionsvorhaben deutlich unterscheiden können.

Um mehr Flexibilität zu erreichen, wird eine Segmentieroption in mehrere Stufen mit unterschiedlichen Kriterien unterteilt, die nacheinander ausgeführt werden. Die gefundenen Segmente einer Stufe dienen dabei als Suchraum der nachgelagterten. So lassen sich komplexere Fälle modellieren.

Dabei kann für jede Stufe separat spezifiziert werden, wie der Segmentierer mit auftretenden Verschachtelungen umgehen soll, beispielsweise wenn anhand von -Tags segmentiert wird und diese innerhalb der Textfassung zur Auszeichnung von Kapiteln, Unterkapiteln und Sektionen genutzt wurden. Verfügbare Optionen hierbei sind entweder die Nutzung der innen oder der außen liegenden Segmente sowie als zusätzliche Möglichkeit das Duplizieren der entsprechenden Abschnitte.

Zudem kann der Umgang mit Token außerhalb der gefundenen Segmentgrenzen spezifiziert werden: Entweder werden diese verworfen oder als eigenständige Segmente gehandhabt.

Bei der Implementierung des Segmentierers wurde auf einen generischen Aufbau geachtet, um neue Segmentieroptionen leicht ergänzen zu können. Eine Segmentieroption ist dabei als kompakte Klasse beschreibbar, die im wesentlichen (für jede Stufe) das Kriterium zur Grenzziehung enthält sowie die Spezifizierung der Interpretation der zurückgelieferten Grenzen (Startpunkt, Endpunkt, Bereich), des Umgangs mit Verschachtelung (innen, außen, duplizieren) und des Umgangs mit Token außerhalb der gefundenen Segmente (verwerfen, eigenständige Segmente).

Neben projektspezifischen Segmentieroptionen, die von Entwickler:innen programmiert und (derzeit noch) im Quellcode über eine Programmierschnittstelle ergänzt werden müssen, steht den Nutzer:innen von LERA eine Auswahl von 15 vordefinierten Optionen für allgemeine Segmentgrenzen, wie den erwähnten Zeilen, Sätzen und Absätzen, sowie für genrespezifische Segmentgrenzen, beispielsweise für Gedichte oder Dramen, bereit.

5.3 Vergleich auf Makroebene – Die Alignierung von Textsegmenten

Der zweite Schritt, beziehungsweise die erste Vergleichsstufe, zum Bestimmen der Textvarianten ist eine Alignierung der Textsegmente. Dieses Unterkapitel stellt das Problem zunächst formal dar und zeigt die zu lösenden Schwierigkeiten beim Vergleich auf Makroebene auf. Im darauf folgendem Abschnitt wird mit dem Chain-Graph-Signature-Aligner die in LERA verwendete Heuristik kurz vorgestellt, mit der die Textsegmente einander automatisch anhand ihrer Ähnlichkeit zugeordnet werden können. Abschnitt 5.3.3 stellt die beiden anderen in LERA verfügbaren Alignierungsoptionen vor.

5.3.1 Problembeschreibung

Gegeben sei eine Folge aus $n \geq 2$ Textfassungen $F_1, \ldots, F_n$, welche in jeweils $n_i \geq 1$ Textsegmente $S_{i,1}, \ldots, S_{i,n_i}$ zerlegt wurden. Die Textsegmente besitzen dabei eine feste Reihenfolge, sodass die ursprüngliche Textfassung durch die Konkatenation ihrer Textsegmente wieder zusammengesetzt werden kann, das heißt $F_i = \sum_{j=1}^{n_i} S_{i,j}$. Ohne Beschränkung der Allgemeinheit liegt auch für die Textfassungen eine gegebene Reihenfolge vor, was die folgende Modellierung erleichtert.
Gesucht wird nun eine *Alignerung*, das heißt eine Zuordnung ähnlicher Textsegmente über die Fassungen hinweg. Diese kann als Tabelle A modelliert werden, welche pro Textfassung F_i eine Spalte $A_{i,\_}$ sowie m Zeilen besitzt. Die Zellen $A_{i,k}$ einer Spalte i mit $k \in \{1, \ldots, m\}$ entsprechen dabei entweder einem der Textsegmente $S_{i,j}$ der dazugehörigen Fassung F_i oder sind leer (ϵ). Es wird gefordert, dass $\forall i \in \{1, \ldots, n\} : F_i = \sum_{k=1}^{m} A_{i,k}$ gilt, das heißt, die ursprüngliche Reihenfolge der Segmente in ihrer vertikalen Anordnung erhalten bleibt und die Alignierung somit die ursprüngliche Leserichtung widerspiegelt. Durch das Einfügen leerer Zellen (ϵ) entsteht die eigentliche Alignierung in den Zeilen der Tabelle: Die Segmente innerhalb einer Zeile wurden aufgrund ihrer Ähnlichkeit einander zugeordnet.

Daraus ergeben sich zwei wesentliche Fragen für die Konstruktion einer Alignierung. Zum einen: Wann sind Textsegmente *ähnlich* und können aligniert werden? Und zum anderen: Was ist eine gute, oder gar optimale Alignierung bezogen auf die ganze Länge der Textfassungen?

Der ersten Frage kann sich mathematisch mit Ähnlichkeits- beziehungsweise Distanzmaßen wie dem Jaccard-Koeffizient [73], der Levenshtein-Distanz [85] oder der Word Movers Distanz [83] angenähert werden. Wie hoch der Grad der Übereinstimmung sein muss, damit zwei Textsegmente als *ähnlich* angesehen werden können, ist allerdings eine subjektive Entscheidung[1]. Erfahrungswerte aus den begleiteten Editionsprojekten bestätigen diesen Eindruck insofern, dass sich die Meinungen von Editor:innen im Einzelfall mitunter unterscheiden. Auch spielen die Segmentstruktur und der Grad der Varianz zwischen den Textfassungen eine wesentliche Rolle. Bei sehr ähnlichen Textfassungen mit langen Segmenten unterscheiden sich die Maßzahlen deutlich von einem sehr variantenreichen Text mit kurzen Segmenten. So ist es in der Regel nicht möglich, einen festen Schwellwert anzugeben und darüber ein finales Ergebnis zu erzielen.

Die zweite Frage – die Güte einer Alignierung zu bewerten – ist ebenfalls von subjektiven Präferenzen der Editor:innen abhängig, aber auch im Kern nicht trivial. Eine Alignierung birgt grundlegend den Zielkonflikt, möglichst ähnliche Textsegmente zu alignieren und gleichzeitig möglichst viele Übereinstimmungen zwischen Textsegmenten abzubilden. Durch die geforderte Beibehaltung der Reihenfolge der Textsegmente sind beide Ziele im Allgemeinen nicht gleichzeitig zu erreichen. Das Problem wurde in [33] über die Modellierung als lineares Optimierungsproblem für genau zwei Textfassungen formalisiert. Die Modellierung ist dabei auf beliebige Ähnlichkeitsmaße übertragbar. Mittels einer Konstante ist zudem steuerbar, ab wann zwei Textsegmente als ähnlich genug zum Alignieren gelten. Das Problem gestaltet sich ähnlich bei der Übertragung auf mehr als zwei Textfassungen, allerdings mit einem erheblichen Zuwachs an Komplexität[2].

Entsprechend LERAs Konzeption soll der Vergleich auf Makroebene die Textfassungen als gleichrangig betrachten. Das heißt, dass bei der Alignierung nach Über-

[1] Siehe auch [38]: „[…] the assessment of findings in the Humanities [and not only] is based on interpretation. While it certainly can be supported by computational means, it is not necessarily computable. As a concrete consequence of that difference in methodology, CollateX offers its users not one algorithm optimized by specific criteria, but a choice between several alignment algorithms so they can select the one that supports their expected results best, always assuming that any computational heuristic may fail in the light of subjective judgement.“.

[2] Die mathematische Modellierung einer optimalen Alignierung für beliebig viele Textfassungen analog zu [33] ist eine aktuelle Forschungsfrage als Teil einer anderen Dissertation und wird hier daher nicht verfolgt.

einstimmungen zwischen den Segmenten aller Textfassungen gesucht werden, basierend auf ihrer Ähnlichkeit als dem wichtigsten Kriterium und nicht der Anordnung der Textfassungen. Daraus ergibt sich schnell ein Effizienzproblem, da die Ähnlichkeit paarweise zwischen allen Textsegmenten aus unterschiedlichen Textfassungen überprüft werden muss. Formal wären das bei n Textfassungen mit je n_i Textsegmenten:

$$\sum_{i=1}^{n-1}(n_i \cdot \sum_{j=i+1}^{n} n_j) \text{ Vergleiche.}$$

Im Fall der vier betrachteten Fassungen des sechsten Buchs der *Histoire des deux Indes* mit 246, 273, 390 und 421 Absätzen sind das beispielsweise 652.257 Vergleiche. Das sind zu viele für eine Echtzeitberechnung, gerade in einer interaktiven Anwendung wie LERA, bei der schnelle Reaktionszeiten des Systems erwartet werden. Ein vollständiger paarweiser Vergleich der Textsegmente für die Bestimmung einer optimalen Alignierung ist daher für den allgemeinen Anwendungsfall nicht durchführbar.

5.3.2 Chain-Graph-Signature-Aligner

Für den Vergleich auf Makroebene steht in LERA eine automatische Heuristik bereit, die Textsegmente einander anhand ihrer Ähnlichkeit zuordnet. Der verwendete Ansatz geht maßgeblich auf die Arbeit von André Medek (né Gießler) im Projekt *Semi-automatische Differenzanalyse von komplexen Textvarianten (SaDA)* zurück und wird in [118] detailliert beschrieben. Im Folgenden wird die Grundidee der Heuristik vorgestellt.

Grundlegend wird ein mathematisches Ähnlichkeits- beziehungsweise Distanzmaß benötigt, um die Übereinstimmung von zwei Textsegmenten von einem automatischen Ansatz bewerten lassen zu können und sie in der Folge gegebenenfalls zu alignieren. In der in LERA verwendeten Heuristik wird dafür der Jaccard-Koeffizient [73] genutzt, wobei die Heuristik prinzipiell auch die Verwendung anderer sowie die Kombination aus mehreren Maßzahlen erlaubt. Der Jaccard-Koeffizient für zwei Textsegmente S_A und S_B ergibt sich als:

$$JC(A, B) = \frac{|A \cap B|}{|A \cup B|},$$

wobei A und B die Mengen der in S_A und S_B vorkommenden Token bezeichnen.

Der Jaccard-Koeffizient entspricht also dem Anteil des Gesamtvokabulars beider Textsegmente, der tatsächlich in beiden Segmenten vorkommt.

Zur Steigerung der Effizienz und Effektivität der Heuristik werden zunächst Signaturen der Textsegmente berechnet. Im Falle der Anwendung des Jaccard-Koeffizienten ist das beispielsweise die Reduktion der Tokenmengen auf signifikante Token. In LERA wird dazu auf die vorliegenden Tokenklassen zurückgegriffen, sodass nur Worte oder Zahlen in die Betrachtung einfließen. Optional kann zudem eine Mindestlänge an Zeichen festgelegt werden, um so heuristisch Stoppwörter auszuschließen. Weitere Faktoren sind denkbar, wie die Reduktion auf bestimmte Wortarten oder die Wichtung der Wörter über ein Bewertungsmaß[3].
Im Anschluss findet intern eine Normalisierung der Token mittels vorkonfigurierter Filter statt, was die Umwandlung von Groß- in Kleinbuchstaben und das Entfernen von kombinierenden Zeichen beinhaltet.
Um einen vollständigen, paarweisen Vergleich zwischen allen Segmenten aller Textfassungen zu vermeiden, wird die Ähnlichkeitsberechnung auf Paare eingeschränkt, für die es entsprechend ihrer Signatur überhaupt sinnvoll erscheint. Bei Anwendung des Jaccard-Koeffizienten, kann die Berechnung zum Beispiel auf Paare beschränkt werden, bei denen mindestens ein signifikantes Wort in beiden Textsegmenten vorkommt. Um das zu erreichen, wird im Rahmen der Heuristik das gesamte Vokabular bestimmt und ein Index der Vorkommen signifikanter Wörter aufgebaut. So kann die tatsächliche Berechnung des Jaccard-Koeffizienten auf aussichtsreiche Paare von Textsegmenten beschränkt werden. Im Fall von Buch 6 der *Histoire des deux Indes* reduziert sich damit die Anzahl der Vergleiche je nach Parameterwahl von 652.257 auf deutlich unter 10.000, sodass die absatzweise Alignierung in unter einer halben Minute berechnet werden kann[4].

Neben den Signaturen ist die gewählte Datenstruktur der Kettengraphen namensgebend für die Heuristik. Textfassungen werden dabei initial als voneinander isolierte, gerichtete Graphen modelliert. Die Textsegmente bilden die Knoten, wobei aufein-

[3] Beispielsweise das TF-IDF-Maß, welches in Abschnitt 6.3.1 im Kontext der in LERA integrierten Wortwolken diskutiert wird.

[4] Die Laufzeit betrug für zehn Messungen im Mittel 29,63 Sekunden bei einer Standardabweichung von 0,23 Sekunden. Getestet wurde dabei auf einem Laptop mit einem 64-bit-Betriebssystem (Ubuntu 22.04), vier Kernen mit einer Taktung von je 2,40GHz (Intel® Core™ i7-5500U) und acht GiB DDR3-Arbeitsspeicher mit einer Geschwindigkeit von 1600 MT/s.

anderfolgende Segmente durch eine gerichtete Kante verbunden werden. Die Heuristik aligniert zwei Textsegmente nun, indem die beiden dazugehörigen Knoten in einem vereinigt werden, wobei die Kanten entsprechend umgebogen werden. In der Folge werden auch die zuvor isolierten Kettengraphen verbunden. Dabei wird in der Heuristik darauf geachtet, dass die zu vereinigenden Knoten keine sich überkreuzenden Kanten besitzen, da in der daraus abgeleiteten Alignierung die Reihenfolge der Textsegmente in mindestens einer Textfassungen durcheinander gebracht werden würde. Die Alignierungsheuristik muss dazu das Entstehen von Zyklen im Graphen erkennen und verhindern[5]. Ungültige Paare zeigen dabei die Transposition von Textsegmenten auf. Diese werden zwar nicht aligniert, können aber als ergänzendes Ergebnis von der Heuristik zurück geliefert und so von LERA visualisiert werden.

Die Heuristik wählt als Nächstes stets das Paar von Textsegmenten mit der größten Ähnlichkeit und prüft, ob es dieses alignieren kann. Aufgrund dieser gierigen (greedy) Vorgehensweise handelt es sich beim berechneten Ergebnis in der Regel nicht um eine optimale Alignierung nach [33]. Zwar werden mit dem Verfahren stets möglichst ähnliche Textsegmente aligniert, doch führt diese Bevorzugung einzelner Alignierungskanten mitunter dazu, dass gegebenenfalls mehrere ebenfalls gute Alignierungskanten verworfen werden müssen. Zudem wird bei zwei oder mehr gleich guten Optionen automatisch eine Kante vom Algorithmus gewählt, was wiederum von der Eingabereihenfolge der Textfassungen abhängen kann. Der Algorithmus arbeitet somit zwar gleichrangig, aber nicht vollständig unabhängig von der vorgegebenen Reihenfolge der Textfassungen.

Aufgrund dieser Umstände sowie der oben diskutierten Subjektivität bei der Bewertung der Güte einer Alignierung wird in LERA ein semi-automatischer Ansatz verfolgt. Mittels eines anpassbaren Schwellwerts für das verwendete Ähnlichkeitsmaß wird zunächst ein möglichst guter, automatisch berechneter Vorschlag für die Alignierung gemacht. Die finale Entscheidung liegt aber bei den Editor:innen, die über die grafische Nutzeroberfläche die Alignierung (sowie Segmentierung) nachträglich den eigenen Vorstellungen anpassen können.

[5] André Medek hat die Gültigkeitsprüfung durch die Modellierung als Kettengraphen und den Einsatz balancierter binärer Suchbäume in $O(log(n) + k)$ lösen können, wobei k die Anzahl der Ketten/Textfassungen und n die Anzahl aller Knoten/Textsegmente ist.

5.3.3 Weitere Alignierungsoptionen in LERA

Neben der automatischen Alignierung auf Basis der Ähnlichkeiten stehen in LERA zwei alternative Alignierungsoptionen in der Nutzeroberfläche zur Verfügung.

Zum einen ist dies die Alignierung in der vorgegebenen Reihenfolge der Segmente. Diese Option kann hilfreich sein, wenn die Textsegmente sehr unterschiedlich sind, die Textfassungen aber eine identische beziehungsweise sehr ähnliche Struktur aufweisen und dadurch entsprechend wenig manuell eingegriffen werden muss.
Ein Anwendungsbeispiel hierfür sind die Übersetzungen eines Textes in unterschiedlichen Sprachen, die den Nutzer:innen synoptisch einander gegenübergestellt präsentiert werden sollen, ohne eine detaillierte Differenzanalyse vom System vornehmen zu lassen.

Zum anderen existiert die Möglichkeit, Textsegmente mit einem übereinstimmenden *xml:id*-Attribut einander zuzuordnen. Diese Angabe wird dazu zuvor automatisch vom System aus den hochgeladenen XML-Dateien der Textfassungen extrahiert und in der Datenstruktur ihrer Segmente vermerkt. Somit kann externes Wissen in die Alignierung eingebracht werden, was beispielsweise für Editionsvorhaben relevant ist, bei denen die Textfassungen manuell durch Expert:innen aligniert werden.

Der Prozess der Alignierung wurde in LERA darüber hinaus stark modularisiert, sodass weitere Methoden zur Alignierung leicht eingebunden werden können. So lässt sich im Rahmen eines konkreten Editionsvorhabens auch werks- oder domänenspezifisches Wissen in eine automatische Heuristik einbetten und als weitere Option über LERAs Nutzeroberfläche zur Verfügung stellen.
Ein Beispiel hierfür ist die Edition des *Manual de confesores y penitentes*, bei der verschiedensprachige Textfassungen eines mittelalterlichen Rechtstextes untersucht und mittels einer projektinternen, in LERA integrierten Heuristik aligniert werden. In diese fließen unterschiedliche Kriterien ein, wie identische Bezeichner in den Marginalien – oft vom Autor in spätere Textfassungen übernommen – oder auch das Übereinstimmen von n-Grammen auf Buchstabenebene, die zwischen den portugiesischen und spanischen Textfassungen bei ähnlichem Inhalt vermehrt auftreten [148].

5.4 Vergleich auf Mikroebene – Die detaillierte Differenzanalyse

Der dritte Schritt beim Textvergleich hat das Ziel, die Gemeinsamkeiten und Unterschiede zwischen den alignierten Segmenten auf Wort- beziehungsweise Tokenebene zu erkennen und für die Darstellung in der Nutzeroberfläche aufzubereiten. In LERA wird dabei standardmäßig ein gleichberechtigter Vergleich aller Textfassungen verfolgt, bei dem die Textvarianten samt der korrespondierenden Textpassagen in allen Fassungen hervorgehoben werden.
Dieses Unterkapitel beschreibt den in LERA umgesetzten Ansatz zum gleichrangigen Textvergleich auf Mikroebene im Detail und gliedert sich wie folgt. Zunächst folgt eine formale Definition der Begriffe Textvergleich, Textvariante und Textgemeinsamkeit sowie eine Einordnung der Güte eines Textvergleichs. Daraus geht hervor, dass es sich beim gleichrangigen Textvergleich um ein Optimierungsproblem handelt. Eine Optimallösung zu bestimmen, ist dabei NP-schwer, wie in Abschnitt 5.4.2 über die Reduktion des bekannten NP-schweren Problems des Auffindens einer maximal stabilen Knotenmenge in einem Graphen auf das Textvergleichsproblem gezeigt wird. Zudem folgt der Beweis der NP-Vollständigkeit des Problems. Um trotz der bewiesenen Komplexität einen gleichrangigen Textvergleich in LERA zu realisieren, wird schließlich in Abschnitt 5.4.3 eine Heuristik auf Basis der Levenshtein-Distanz vorgestellt sowie ihre Laufzeit und zusätzliche für LERA implementierte Erweiterungen diskutiert.

5.4.1 Formale Definition eines Textvergleichs

Gegeben sei eine Menge $\mathbb{S}$ bestehend aus $n \geq 2$ Textsegmenten $S_1, \ldots, S_n$, welche wiederum jeweils aus $m_i \geq 1$ Token bestehen, das heißt $S_i = \sum_{j=1}^{m_i} T_{i,j}$.
Sei $\mathbb{T} = \{T_{i,j} : 1 \leq i \leq n \text{ und } 1 \leq j \leq m_i\}$ die Menge der in den n Textsegmenten enthaltenen Token. Sei ϵ das **leere Token**, welches nicht in $\mathbb{T}$ enthalten ist, das heißt $\epsilon \notin \mathbb{T}$.

Unter einem **Textvergleich** von $\mathbb{S} = S_1, \ldots, S_n$ versteht man eine endliche Folge $VGL = \{VGL_1, \ldots, VGL_{|VGL|}\}$ von Tupeln mit

$$VGL_j \in (\mathbb{T} \cup \{\epsilon\})^n \setminus \{\epsilon\}^n \text{ mit } \forall i \in \{1, \ldots, n\} : S_i = \sum_{j=1}^{|VGL|} VGL_{j,i} \,. \quad (5.1)$$

Ein Tupel VGL_j aus der Folge VGL heißt **Textvariante**, wenn

$$\exists i_1, i_2 \in \{1, \ldots, n\} \text{ mit } i_1 \neq i_2 : VGL_{j,i_1} \neq VGL_{j,i_2}$$

gilt. Ansonsten heißt das Tupel VGL_j **Textgemeinsamkeit**.

Eigenschaften eines Textvergleichs

Mit der Modellierung der VGL_j als Tupel der Größe n wird der gleichrangige Vergleich der n Segmente verfolgt. Und zwar der Gestalt, dass zu jedem Token stets auch die korrespondierenden Token in den übrigen Textfassungen gespeichert werden können. Jedes Element $VGL_{j,i}$ entspricht dabei jeweils genau einem Token von S_i oder ϵ. Für Textgemeinsamkeiten gilt:

$$\forall i \in \{1, \ldots, n-1\} : VGL_{j,i} = VGL_{j,i+1}$$

Eine wichtige Eigenschaft der Modellierung von VGL ist, dass die Reihenfolge der Token für jedes Segment gewahrt bleibt, da $S_i = \sum_{j=1}^{|VGL|} VGL_{j,i}$.

Die Modellierung wird im Folgenden an einem Beispiel veranschaulicht. Hierbei sind $n = 3$ Textsegmente mit $S_1 = abd$, $S_2 = abcd$ und $S_3 = acd$ gegeben. Ein möglicher Textvergleich entsprechend der Modellierung wäre $VGL = ((a, a, a), (b, b, \epsilon), (\epsilon, c, c), (d, d, d))$, oder als Tabelle veranschaulicht:

	VGL_1	VGL_2	VGL_3	VGL_4
S_1	a	b		d
S_2	a	b	c	d
S_3	a		c	d
	g	v	v	g

In dieser tabellarischen Darstellung werden die verglichenen Segmente jeweils als Zeilen und die Tupel des Textvergleichs jeweils als Spalte repräsentiert. Die leeren Zellen der Tabelle repräsentieren das leere Token ϵ. Eine zusätzliche Zeile am Ende der Tabelle gibt an, ob das jeweilige Tupel eine Textvariante (v) oder -gemeinsamkeit (g) ist.
Der obige Textvergleich hat genauso viele Tupel wie das größte Segment Token besitzt und zudem enthalten auch die Textvarianten mit Ausnahme von ϵ nur identische

Token. Das ist allerdings nach der bisherigen Modellierung entsprechend der Definition 5.1 keine Notwendigkeit. Ein korrekter Textvergleich für das Beispiel wäre auch:

	VGL_1	VGL_2	VGL_3	VGL_4
S_1	a	b		d
S_2	a	b	c	d
S_3	a	c		d
	g	v	v	g

In der tabellarischen Darstellung wirkt dieser Textvergleich schlechter als der erste, durch die sich unterscheidenden Token b und c im Tupel VGL_2. Tatsächlich wären beide Textvergleiche für die praktische Nutzung in LERA identisch, würden in einer Darstellung für die Nutzer:innen lediglich die Token in Textvarianten eingefärbt (und jene in den Gemeinsamkeiten normal dargestellt) werden.
Allerdings ist selbst folgender Textvergleich, bei dem jeder Token als eine eigenständige Textvarianten definiert und entsprechend eingefärbt werden würde, gültig nach Definition 5.1:

	VGL_1	VGL_2	VGL_3	VGL_4	VGL_5	VGL_6	VGL_7	VGL_8	VGL_9	VGL_{10}
S_1	a	b	d							
S_2				a	b	c	d			
S_3								a	c	d
	v	v	v	v	v	v	v	v	v	v

Es stellt sich also die Frage, wie sich die Güte eines Textvergleichs bewerten lässt und wie ein optimaler Textvergleich aussehen sollte.

Ein optimaler Textvergleich

Es gibt verschiedene mögliche Kriterien, um einen Textvergleich zu bewerten (beispielsweise die Anzahl der ϵ). Eine grundlegende Festlegung, die bereits durch die Definition getroffen wurde, ist, dass Unterschiede zwischen den Textsegmenten niemals den Textgemeinsamkeiten, sondern stets den Textvarianten zugeordnet (und entsprechend markiert) werden. Es wird also bei einem korrekten Textvergleich keine Textvariante übersehen. Gleichzeitig können laut der bestehenden Definition aber

zu viele Textvarianten auftreten, die auch in allen Fassungen übereinstimmende Token umfassen, wie im Beispiel oben veranschaulicht. Daraus leitet sich folgende Vorgabe ab, die auch anekdotenhaft von einer Kollegin aus einem gemeinsamen Editionsprojekt geäußert wurde („je weniger Farbe, desto besser“): Eine guter Textvergleich sollte alle Unterschiede zwischen den Texten erkennen, aber dabei möglichst wenige Token als Textvariante markieren.

Bezogen auf die Definition des Textvergleichs, sollte demnach die Anzahl der Token in Textvarianten minimiert werden und im Umkehrschluss die Anzahl der Token in Textgemeinsamkeiten maximiert werden. Zur Formalisierung werden die beiden folgenden Maße eingeführt:

$$|VGL|_{var} = \sum_{j=1}^{|VGL|} \begin{cases} 1, & VGL_j \text{ ist eine Textvariante} \\ 0, & VGL_j \text{ ist eine Textgemeinsamkeit} \end{cases}$$

und

$$|VGL|_{gem} = \sum_{j=1}^{|VGL|} \begin{cases} 0, & VGL_j \text{ ist eine Textvariante} \\ 1, & VGL_j \text{ ist eine Textgemeinsamkeit} \end{cases}$$

Die Anzahl der Token in Textgemeinsamkeiten entspricht $n \cdot |VGL|_{gem}$, sodass zur Optimierung eines Textvergleichs direkt die Anzahl der Gemeinsamkeiten $|VGL|_{gem}$ maximiert werden kann. Formalisiert leitet sich daraus folgende Definition ab.

Definition 1 (Optimaler Textvergleich) *Sei $\mathbb{VGL}$ die Menge aller Textvergleiche für eine gegebene Menge von Textsegmenten. Ein Textvergleich $OPT \in \mathbb{VGL}$ heißt **optimal**, genau dann wenn $\nexists VGL \in \mathbb{VGL} : |VGL|_{gem} > |OPT|_{gem}$.*

Aus der obigen Formalisierung des gleichrangigen Textvergleichs auf Tokenebene ergibt sich ein Optimierungsproblem, das wie folgt definiert ist.

Definition 2 (Textvergleichs-Optimierungsproblem) *Sei $\mathbb{S}$ eine Menge von Textsegmenten. Finde k_{opt} mit $k_{opt} = |OPT|_{gem}$ und OPT ist ein optimaler Textvergleich von $\mathbb{S}$.*

Zudem werden im Folgenden das dazugehörige Entscheidungsproblem für die sich im nächsten Abschnitt anschließende Komplexitätsanalyse sowie das dazugehörige Suchproblem für die spätere Realisierung in LERA definiert.

Definition 3 (Textvergleichs-Entscheidungsproblem) *Seien k eine natürliche Zahl und $\mathbb{S}$ eine Menge von Textsegmenten. Gibt es einen Textvergleich mit k Textgemeinsamkeiten für $\mathbb{S}$?*

Definition 4 (Textvergleichs-Suchproblem) *Sei $\mathbb{S}$ eine Menge von Textsegmenten. Finde einen optimalen Textvergleich von $\mathbb{S}$.*

Abgrenzung zum multiplen Sequenzalignment

Das aufgezeigte Textvergleichsproblem ist verwandt mit dem in Abschnitt 3.1.1 diskutierten Problem des multiplen Sequenzalignments, unterscheidet sich aber in einem wesentlichen Punkt. Verfahren im Bereich des Sequenzalignments optimieren meist eine Bewertungsfunktion, die beispielsweise auf Editierdistanzen basieren kann. Dabei wird versucht, zwei Sequenzen mit möglichst wenigen beziehungsweise kostengünstigen Operationen ineinander zu überführen. Dieses Prinzip wird beim multiplen Sequenzalignment auf mehrere Sequenzen übertragen.
Beim aufgezeigten Textvergleichsproblem kommt durch die Umsetzung des gleichrangigen Vergleichs der Textfassungen eine zusätzlich zu erfüllende Bedingung hinzu. Nämlich, dass alle Sequenzen übereinstimmen müssen, um positiv in die zu optimierenden Kosten einzufließen. Das heißt, hierbei wird die Anzahl der *Matches* maximiert, unter der Nebenbedingung, dass ein Match nur dann vorliegt, wenn die Segmente in allen Textfassungen an dieser Stelle gleich sind. Aus diesem Grund, abseits der unterschiedlichen Anwendungsdomäne, sind bestehende Verfahren des multiplen Sequenzalignments nicht auf das Textvergleichsproblem übertragbar.
Diese Modellierung des gleichrangigen Vergleichsprinzips ist zudem maßgeblich für die hohen Komplexität des Textvergleichsproblems, wie in den kommenden Abschnitten gezeigt wird.

5.4.2 Komplexität des Textvergleichsproblems

In diesem Abschnitt wird bewiesen, dass das Textvergleichs-Entscheidungsproblem NP-vollständig ist. Dazu wird zunächst gezeigt, dass es in der Komplexitätsklasse NP liegt. Anschließend folgt der Beweis der NP-Schwere über die Reduktion des Stabilitätsproblems auf das Textvergleichs-Entscheidungsproblem. Durch die Erfüllung beider Bedingungen ist bewiesen[6], dass das Textvergleichs-Entscheidungspro-

[6] Siehe zum Beispiel Lemma 2.3 in [51, S.38].

blem NP-vollständig ist. Daraus ergibt sich wiederum, dass das dazugehörige Optimierungsproblem sowie das dazugehörige, für die Umsetzung in LERA relevante Suchproblem NP-schwer sind.

Theorem 1 *Das Textvergleichs-Entscheidungsproblem ist NP-vollständig.*

Textvergleichs-Entscheidungsproblem liegt in *NP*

Damit ein Problem zur Komplexitätsklasse NP gehört, muss eine gültige Lösung des Problems zum einen stets eine polynomielle Größe aufweisen (I) und zum anderen ihre Gültigkeit stets in polynomieller Zeit verifiziert werden können (II).

Die erste Bedingung (I) ist gegeben, da die Anzahl der Tupel, das heißt Textvarianten und -gemeinsamkeiten, für einen gültigen Textvergleich von $\mathbb{S}$ durch die aufsummierte Anzahl von Token der Segmente aus $\mathbb{S}$ beschränkt wird. Dies folgt direkt aus der Definition eines Textvergleichs, die zum einen Tupel verbietet, die nur das leere Token ϵ enthalten und zum anderen sicher stellt, dass genau die gleiche Anzahl von Token in den Tupeln vorkommt, wie es sie in den Segmenten gibt.

Die zweite Bedingung (II) wird durch den folgenden Verifikationsalgorithmus erfüllt, der die Gültigkeit eines gegebenen Textvergleichs für $\mathbb{S}$ in polynomieller Zeit prüfen kann.
Sei VGL ein Textvergleich für die Segmentmenge $\mathbb{S} = S_1, \ldots, S_n$. Zunächst wird verifiziert, dass die Tupel $VGL_j \in VGL$ mit $j \in \{1, \ldots, |VGL|\}$ überkreuzungsfrei sind, indem die n Bedingungen aus Definition 5.1 geprüft werden. Demnach muss gelten: $\forall i \in \{1, \ldots, n\} : S_i = \sum_j^{|VGL|} VGL_{j,i}$. Die Anzahl der notwendigen Berechnungsschritte ist dabei durch die Anzahl der Segmente und der Tupel beschränkt, welche wiederum nach der ersten Bedingung (I) durch die Anzahl der Token beschränkt ist.
Zudem muss verifiziert werden, dass jedes Tupel mindestens einen Token ($\neq \epsilon$) enthält. Dies geschieht, indem die Elemente der Tupel durchlaufen und auf Ungleichheit zu ϵ geprüft werden. Sobald ein Token $T \neq \epsilon$ entdeckt wird, gilt das Tupel als zulässig. Existiert ein unzulässiges Tupel wird der Textvergleich als ungültig zurückgewiesen.
Zuletzt wird die Anzahl an Textgemeinsamkeiten ermittelt. Dazu werden die Tupel erneut einzeln durchlaufen und jedes Tupel in eine Menge von Token umgewandelt.

Besitzt solch eine Menge Kardinalität 1, handelt es sich um eine Textgemeinsamkeit, da alle Token des dazugehörigen Tupels übereinstimmen und durch den vorherigen Überprüfungsschritt sichergestellt ist, dass es sich dabei nicht um ϵ handelt. Ist die Anzahl an Textgemeinsamkeiten mindestens so groß wie k, wird die Lösung als gültig akzeptiert.

Da beide Bedingungen erfüllt sind, liegt das Textvergleichs-Entscheidungsproblem in NP.

Textvergleichs-Entscheidungsproblem ist *NP*-schwer

Um zu zeigen, dass das Textvergleichs-Entscheidungsproblem auch NP-schwer ist, wird im Folgenden das NP-vollständige Stabilitätsproblem (auch als Co-Clique oder unabhängige Menge bezeichnet) mittels Polynomialzeitreduktion auf das Textvergleichs-Entscheidungsproblem abgebildet.

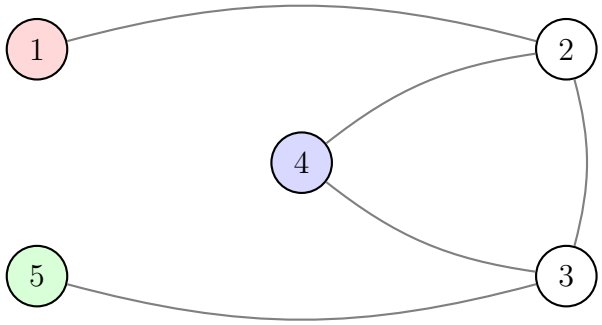
Definition 5 (Stabilitätsproblem) *Gegeben sei eine natürliche Zahl k und ein einfacher Graph $G = (V, E)$. Die Menge $V' \subset V$ heißt stabile Menge für G, genau dann wenn $\nexists (v, v') \in E : v \in V' und\ v' \in V'$. Das Stabilitätsproblem fragt: Existiert eine stabile Menge der Kardinalität k für G?*

Eine beliebige Instanz des Stabilitätsproblems (k, G) kann in polynomieller Zeit auf das Textvergleichs-Entscheidungsproblem abgebildet werden. Die Konstruktion geht wie folgt:
Sei $G = (V, E)$ ein einfacher Graph (ein ungerichteter Graph ohne Schleifen und Multikanten) mit $n = |V|$ und $m = |E|$ sowie k eine natürliche Zahl. Die Knoten seien durchnummeriert beginnend bei 1.
Die Menge der Token wird als Menge der ersten n natürlichen Zahlen modelliert, wobei jedes Token einen der n Knoten repräsentiert. Die Token lassen sich durch diese Modellierung nach einer festen Reihenfolge ordnen.
Nun werden $2 \cdot m + 1$ Textsegmente konstruiert. Zunächst wird ein initiales Segment $S_0 = 1, \ldots, n$ erzeugt. Für jede Kante $e_i = (a, b) \in E$ zwischen den Knoten – repräsentative Token – a und b mit o.B.d.A. $a < b$ werden zwei Segmente $S_{2 \cdot i-1}$ und $S_{2 \cdot i}$ erzeugt. $S_{2 \cdot i}$ ist eine exakte Kopie von S_0 [7].

[7] Die Segmente $S_{2 \cdot i}$ sind nicht zwingend notwendig für die Konstruktion, erleichtern aber das Verständnis für den nachstehenden Beweis. So lässt sich mit ihnen leichter nachvollziehen, dass die Reihenfolge der Token stets gewahrt bleibt.

Zur Konstruktion des Segments $S_{2 \cdot i-1}$ werden die Textpassagen $x = \sum_{j=1}^{a-1} j$, $y = \sum_{j=a+1}^{b-1} j$, und $z = \sum_{j=b+1}^{n} j$ als Konkatenation von Token definiert. Das Segment ergibt sich dann als Konkatenation $S_{2 \cdot i-1} = xybayz$. Dabei bleiben alle Token vor a – das heißt die Textpassage x – am Anfang und analog alle Token nach b in ihrer Reihenfolge am Ende des Segments. Die Token zwischen a und b werden wiederholt und a und b schließlich in ihrer Reihenfolge vertauscht dazwischen platziert.

Folgendes Beispiel illustriert die Konstruktion für einen Graphen mit fünf Knoten und Kanten:



	S_0	1	2	3	4	5	
$(1,2)$	S_1	2	1	3	4	5	
	S_2	1	2	3	4	5	
$(2,3)$	S_3	1	3	2	4	5	
	S_4	1	2	3	4	5	
$(3,4)$	S_5	1	2	4	3	5	
	S_6	1	2	3	4	5	
$(2,4)$	S_7	1	3	4	2	3	5
	S_8	1	2	3	4	5	
$(3,5)$	S_9	1	2	4	5	3	4
	S_{10}	1	2	3	4	5	

Für den Beispielgraphen entstehen durch die Überführung in einen Textvergleich aus den fünf Knoten genau fünf Token $(1, \ldots, 5)$ und aus den fünf Kanten elf Textsegmente. Die farblich hervorgehobene, stabile Menge im Graphen $(\{1, 4, 5\})$ findet sich auch in Form der markierten Textgemeinsamkeiten im konstruierten Textvergleichsproblem wieder.

Die Transformation lässt sich in polynomieller Zeit realisieren, denn es wird nur eine beschränkte Anzahl von $2 \cdot m + 1$ Textsegmenten generiert, von denen jedes maximal $2 \cdot (n-2) + 2$ Token beinhalten kann. Dieser schlechteste Fall tritt auf, wenn $a = 1$ und $b = n$ ist, sodass alle übrigen Token als Textpassage y doppelt platziert werden. Zudem ist m polynomiell in der Anzahl der Knoten beschränkt, da ein einfacher (vollständiger) Graph maximal $½ \cdot n \cdot (n-1)$ Kanten besitzen kann.

Bleibt zu zeigen, dass die Antworten für eine gegebene Instanz des Stabilitätsproblems und die daraus konstruierte Instanz des Textvergleichs-Entscheidungsproblems identische Antworten liefern.

(I) Existiert ein Textvergleich mit k Textgemeinsamkeiten, so existiert auch eine stabile Menge der Größe k im Graphen G.
Angenommen, für die konstruierten Textsegmente existiert ein Textvergleich mit k Textgemeinsamkeiten. Die Textgemeinsamkeiten müssen paarweise unterschiedliche Token – respektive zugeordnete Knoten – besitzen, da das initiale Segment $S_0 = 1, \ldots, n$ jeden Token genau einmal enthält und ein Token nach Definition 5.1 bei einem Textvergleich VGL nicht gleichzeitig mehreren Textgemeinsamkeiten (oder -varianten) zugeordnet werden kann, da sonst die Bedingung $\forall i \in \{1, \ldots, n\} : S_i = \sum_j^{|VGL|} VGL_{j,i}$ verletzt wäre.
Die repräsentierten Knoten von zwei in Textgemeinsamkeiten auftretenden Token a und b können keine Kante (a, b) in G besitzen, da im konstruierten Segment zur Kante die Token a und b nur genau einmal vorkommen und ihre Reihenfolge gegenüber der initialen Reihenfolge getauscht wurde. So kann folglich ebenfalls nach Definition 5.1 nur einer der beiden Token in eine Textgemeinsamkeit aufgenommen werden.
Die Menge der in den Textgemeinsamkeiten auftretenden Token entspricht damit einer stabilen Menge in G und da jede der k Textgemeinsamkeiten ein unterschiedliches Token beinhaltet, hat diese stabile Menge ebenfalls eine Größe von k.

(II) Existiert eine stabile Menge der Größe k in $G = (V, E)$, so existiert auch ein Textvergleich mit k Textgemeinsamkeiten.
Sei $\{v'_1, \ldots, v'_k\}$ eine stabile Menge der Größe k in G mit $v'_t < v'_{t+1}$ für $t = 1, \ldots, k-1$. Zu zeigen ist, dass jedes der konstruierten Textsegmente S_i mit $i =$

$0, \ldots, 2 \cdot |E|$ zerlegbar ist in:

$$S_i = \ldots v'_1 \ldots v'_2 \ldots\ldots v'_t \ldots\ldots v'_k \ldots$$

Für alle geraden i, das heißt $i = 2 \cdot j$ mit $j = 0, \ldots, |E|$, gilt, das Segment S_i wurde als Folge $1, \ldots, |V|$ konstruiert und enthält entsprechend die gewünschte Anordnung. Für alle ungeraden i, das heißt $i = 2 \cdot j - 1$ mit $j = 1, \ldots, |E|$, sei das Segment $S_i = xybayz$ entstanden durch die Kante $(a, b) \in E$ mit $a < b$. So sind drei mögliche Fälle zu prüfen. Für den ersten Fall mit $\{a, b\} \cap \{v'_1, \ldots, v'_k\} = \emptyset$ findet sich die gewünschte Anordnung in der Tokenfolge xyz. Für den zweiten Fall mit $\{a, b\} \cap \{v'_1, \ldots, v'_k\} = \{a\}$ findet sich die gewünschte Anordnung in der Tokenfolge $xayz$. Für den dritten Fall mit $\{a, b\} \cap \{v'_1, \ldots, v'_k\} = \{b\}$ findet sich die gewünschte Anordnung in der Tokenfolge $xybz$. Ein vierter denkbarer Fall mit $\{a, b\} \cap \{v'_1, \ldots, v'_k\} = \{a, b\}$ ist ausgeschlossen, da zwischen a und b eine Kante bestünde und nicht beide gleichzeitig in einer stabilen Menge vorkommen können. Demnach können für alle Knoten aus der stabilen Menge Textgemeinsamkeiten gebildet werden, da sie paarweise überkreuzungsfrei in den konstruierten Segmenten vorkommen.

Aus (I) und (II) folgt, die gelieferten Antworten für eine gegebene Instanz des Stabilitätsproblems und die daraus konstruierte Instanz des Textvergleichs-Entscheidungsproblems sind identisch, was zusammen mit der gezeigten, in polynomieller Zeit realisierbaren Transformation das Textvergleichs-Entscheidungsproblem NP-schwer macht. Da das Textvergleichs-Entscheidungsproblem, wie gezeigt, zudem in NP liegt, ist es NP-vollständig und Theorem 1 bewiesen.

5.4.3 Heuristik zur detaillierten Differenzanalyse

Wie im vorherigen Unterkapitel gezeigt, handelt es sich beim Textvergleichs-Suchproblem um ein NP-schweres Problem. Dennoch muss ihm in LERA praktisch begegnet werden, um die Textvarianten in einer Menge von zuvor alignierten Segmenten $\mathbb{S}$ hervorzuheben. Dabei ist die Effizienz des Verfahrens ein besonders wichtiger Faktor, da LERA eine interaktive Webanwendung ist, die möglichst in Echtzeit auf geänderte Eingaben durch die Nutzer:innen reagieren muss.
Die Größenordnung für $n = |\mathbb{S}|$ bewegt sich dabei meist im Bereich von bis zu acht Textfassungen, welche unter anderem durch die Bildschirmbreite bei der Darstellung als Spaltensynopse limitiert ist. Es gibt aber auch Editionsvorhaben, die diesen

Wert deutlich übersteigen, wie beispielsweise die Editionen zu *Keter Shem Ṭov* und *Kalīla and Dimna*, bei denen Dutzende Textfassungen vorliegen und mitunter für die Untersuchung kollationiert werden. Dabei bleibt n in einer Größenordnung von unter 100.
Zum anderen bewegt sich die Anzahl der Token eines Segments oft im Bereich von zehn bis 200, wenn beispielsweise die Segmentierung nach Sätzen oder Absätzen vorgenommen wurde. Doch auch hier gibt es Ausnahmen. Zum Beispiel wenn ganze Kapitel oder Unterkapitel von Büchern miteinander verglichen werden sollen oder die Textfassungen zuvor gar nicht segmentiert wurden.
Die Größenordnungen sind in der Praxis zwar mitunter relativ klein, aber in der Regel dennoch zu groß, um einen optimalen Textvergleich zu bestimmen. Aus diesem Grund wurde für LERA ein heuristisches Verfahren entwickelt, um die Textvarianten und -gemeinsamkeiten zu bestimmen.

Grundidee der Heuristik

Aufgrund der gezeigten Komplexitätsklasse, ist die direkte, exakte Bestimmung eines optimalen Textvergleichs in der Praxis nicht möglich. Auch bestehende Ansätze aus dem verwandten Problem des multiplen Sequenzalignments, wie die Abgrenzung in Abschnitt 5.4.1 zeigt, können wegen der unterschiedlichen Optimierungskriterien nicht direkt übertragen werden.
Um der Komplexität zu begegnen, könnte der im Kontext von multiplen Sequenzalignments verwendete Ansatz der progressiven Alignierung auf das Textvergleichsproblem übertragen werden. Bei diesem heuristischen Ansatz, der beispielsweise bei CollateX [36] verwendet wird, werden zunächst zwei Textfassungen miteinander verglichen und dann schrittweise die weiteren Fassungen dem Ergebnis hinzugefügt. Dieses Vorgehen scheidet für LERA konzeptionell allerdings aus, denn auch mit der detaillierten Differenzanalyse soll ein gleichrangiger Textvergleich verfolgt werden. Bei der progressiven Alignierung werden früh hinzugefügte Textfassungen bevorzugt behandelt, sodass kein gleichrangiger Vergleich vorliegt.

Letztlich bleibt die Option eines paarweisen Vergleichs, bei dem alle Zwischenresultate gleichrangig im Gesamtergebnis berücksichtigt werden.
Ansätze zum paarweisen Sequenzalignment, wie der Needleman-Wunsch-Algorithmus [101], haben eine (quadratische) Laufzeit in $O(m_i \cdot m_j)$, wobei m_i und m_j die Längen der beiden Sequenzen ist.

Auch die Editierdistanz zwischen zwei Texten kann prinzipiell nicht in stark subquadratischer Zeit berechnet werden, wie 2014 von Backurs und Indyk gezeigt [12].

Dennoch ist eine Heuristik auf Basis der Editierdistanz sinnvoll. Da beim Vergleich von Textfassungen üblicherweise sehr ähnliche Tokenfolgen miteinander verglichen werden, kann hier der Ansatz von Ukkonen [146] zur Bestimmung der Editierdistanz sowie der dazugehörigen Edit-Scripte verwendet werden. Dessen Laufzeit liegt in $O((s+1) \cdot min(m_i, m_j))$, wobei m_i und m_j die Länge der beiden Tokenfolgen und s die Editierdistanz zwischen ihnen ist. Dieser Algorithmus ist besonders schnell für sehr ähnliche Textsegmente und damit niedrige Werte für s, welche in den Anwendungsfällen von LERA häufig auftreten.

Zudem ist es nicht notwendig, die Edit-Scripte zwischen allen Paaren von Textsegmenten zu berechnen. Grundlage ist die Beobachtung, dass zwei Edit-Scripte $ES_{A \Rightarrow B}$ und $ES_{B \Rightarrow C}$, welche zum einen die notwendigen Operationen listen, um A nach B und zum anderen B nach C zu transformieren, auch alle Informationen enthalten, um A nach C zu transformieren. Zwar kann sich das Edit-Script $ES_{A \Rightarrow C}$ deutlich unterscheiden, wenn es direkt berechnet wird, anstatt es als eine nacheinander stattfindende Ausführung von $ES_{A \Rightarrow B}$ und $ES_{B \Rightarrow C}$ betrachtet wird – zum Beispiel bei drei Tokenfolgen $A =$ „ab“, $B =$ „$aXYZb$“ und $C =$ „ab“ –, doch sind in $ES_{A \Rightarrow B}$ und $ES_{B \Rightarrow C}$ alle notwendigen Angaben zur Rekonstruktion der Textgemeinsamkeiten und Textvarianten zwischen A und C enthalten. So genügt es, eine Reihenfolge für die Textsegmente festzulegen und nur aufeinander folgende zu vergleichen. Damit reduziert sich die Anzahl der zu berechnenden Edit-Scripte für n Textsegmente von $½ \cdot n \cdot (n-1)$ auf $n-1$.

Die Grundidee der in LERA verwendeten Heuristik zur detaillierten Differenzanalyse einer gegebenen Folge von Textsegmenten besteht nun darin, zunächst die Edit-Scripte aufeinander folgender Textsegmente zu bestimmen und anschließend alle Edit-Scripte simultan mit einem Sweepline-Ansatz zu durchlaufen: Finden sich nur kopierte Token an den aktuell betrachteten Positionen aller Edit-Scripte, handelt es sich dabei um eine Textgemeinsamkeit und die Sweepline wird zu den nächsten Positionen verschoben. Wenn mindestens ein Token nicht kopiert, sondern hinzugefügt, gelöscht oder ersetzt wurde, wurde eine neue Textvariante gefunden. In diesem Fall werden alle danach mit der Sweepline durchlaufenen Token der Variante zugeschrieben, bis sie wieder ausschließlich auf kopierte Token zeigt.

Basisvariante des Sweepline-Algorithmus

Der im Folgenden vorgestellte Sweepline-Algorithmus erhält als Eingabe eine Folge von n Textsegmenten $\mathbb{S} = S_1, ...S_n$, die wiederum jeweils aus einer Folge von $m_i \geq 1$ Token bestehen, das heißt $S_i = \sum_{j=1}^{m_i} T_{i,j}, \forall i \in \{1, \ldots, n\}$.
Der Algorithmus berechnet dazu einen gültigen Textvergleich nach Definition 5.1 und liefert ihn in Form zweier Datenstrukturen – den Textgemeinsamkeiten G und den Textvarianten V – zurück. Dabei handelt es sich um Folgen, deren Elemente Tupel der Dimension n sind, die auf die gegebenen Textsegmente referenzieren. Für die praktische Umsetzung in LERA werden dabei aufeinander folgende Textgemeinsamkeiten zu einem Element in G zusammengefasst, das heißt, statt Referenzen auf einzelne Token werden Referenzen auf Textpassagen zurück geliefert. Gleiches gilt für die in V aufgeschlüsselten Textvarianten. Eine Referenz beschreibt die referenzierte Textpassage in S_i mittels einer Start- und Endposition im Intervall $[1, m_i]$. Um einzelne Token zu referenzieren, werden Start- und Endposition auf den selben Wert gesetzt. Eine Referenz kann zudem den Wert $^{\mathrm{N}}\upsilon_{\mathrm{L}}$ annehmen, falls im dazugehörigen Textsegment kein Token referenziert werden soll.
Zur Vereinfachung der weiteren Verarbeitung werden die Rückgabewerte G und V als alternierende Folge $G_1, V_1, G_2, \ldots V_{|V|}, G_{|G|}$ generiert, welche die Textsegmente partitionieren. Diese Partitionierung beginnt und endet stets mit einer Textgemeinsamkeit, sodass gilt: $|G| = |V| + 1$. Die Textsegmente können nach dieser Modellierung wieder zusammengesetzt werden, indem die in G und V referenzierten Textpassagen abwechselnd konkateniert werden. Es gilt $S_i = \sum_{j=1}^{|V|}(G_{j,i} + V_{j,i}) + G_{|G|,i}$. Da die einzelnen Referenzen in einer Textgemeinsamkeit $G_{j,i}$ oder Textvariante $V_{j,i}$ auch leer sein können ($^{\mathrm{N}}\upsilon_{\mathrm{L}}$), stellen Randfälle, wie der Anfang oder das Ende eines Textsegments oder Einfügungen und Streichungen, bei dieser Modellierung kein Problem dar.

Zur Arbeit mit Edit-Scripten werden im Folgenden einige Hilfsfunktionen definiert. Ein Eintrag im Edit-Script $ES_{A \Rightarrow B}$ wird dabei als Quintupel der Form $[op, i, T_i, j, T_j]$ mit $op \in \{$*:copy*, *:add*, *:del*, *:sub*$\}$ modelliert. Die Werte i und j referenzieren die Positionen in den Textpassagen A und B, an denen sich die Token T_i und T_j befinden. Entsprechend op gibt es dabei Nebenbedingungen für i, T_i, j und T_j. So gilt bei $op =$ *:copy*, dass $T_i = T_j$ und bei $op =$ *:sub*, dass $T_i \neq T_j$. Bei $op =$ *:add* wird für i der vorherige Wert von i übernommen und $T_i = \epsilon$ gesetzt. Bei $op =$ *:del* analog für j und T_j. Die folgenden Hilfsfunktionen erlauben den Zugriff auf die

einzelnen Komponenten eines Eintrags im Edit-Script ES an der Position idx. Sei $ES(idx) = [op, i, T_i, j, T_j]$, so liefern:

- $ES(idx).op \Rightarrow op$
- $ES(idx).pos\_a \Rightarrow i$
- $ES(idx).pos\_b \Rightarrow j$

Die Basisvariante des Sweepline-Algorithmus arbeitet nun wie folgt. Die zu vergleichenden Textsegmente $\mathbb{S} = S_1, ...S_n$ werden in eingegebener Reihenfolge als Folgen von Token betrachtet. Zwischen benachbarten Textsegmenten wird mithilfe des Algorithmus von Ukkonen [146] jeweils ein minimales Edit-Script bestimmt. Dabei gilt: Edit-Script $ES_i, i \in [1...n-1]$ ist eine Abfolge von Operationen, um das Segment S_i in das darauf folgende Segment S_{i+1} zu überführen.
Nach dem Sweepline-Prinzip werden die Edit-Scripte anschließend in einer Hauptschleife gleichzeitig und schrittweise durchlaufen, bis überall das Ende erreicht wurde. Finden sich überall nur *:copy*-Einträge, rückt der Algorithmus simultan in allen Edit-Scripten eine Position weiter. Wenn nicht, wurde eine neue Textvariante entdeckt und die zuletzt geöffnete Textgemeinsamkeit wird an dieser Position geschlossen.
Die Textvariante wird nun in einer inneren Schleife durchlaufen. Dabei gilt: Solange nicht überall ein *:copy*-Eintrag erreicht wurde, wählt der Algorithmus ein Edit-Script ES_j aus, an dessen aktueller Position ein *:add*, *:del* oder *:sub* steht. Dieses wird nun weiter durchlaufen, bis das nächste *:copy* (oder das Ende) erreicht wurde. Alle dazwischen liegenden Einträge bilden einen Offset, welcher nun auch in allen anderen Edit-Scripten ausgehend vom aktuellen abgetragen wird. Die Reihenfolge der Abarbeitung ist dabei entscheidend, da das Textsegment S_i zunächst in S_{i-1} (beziehungsweise S_{i+1}) überführt werden muss, bevor es mithilfe der entsprechenden Edit-Scripte in Textsegment S_{i-2} (beziehungsweise S_{i+2}) usw. überführt werden kann. Durch das Abtragen wird in allen Textsegmenten die korrespondierende Position des initialen *:copy*-Eintrags von ES_j erreicht. Diese innere Schleife wird solange wiederholt, bis in allen Edit-Scripten ein *:copy*-Eintrag und damit das Ende der aktuellen Textvariante in allen Textsegmenten identifiziert wurde. Damit beginnt wiederum eine neue Textgemeinsamkeit.
Mit dem Terminieren der Hauptschleife, das heißt dem Erreichen des Endes aller Textsegmente, wurden alle Textgemeinsamkeiten ermittelt. Die Textvarianten ergeben sich entsprechend der Modellierung über die Textpassagen dazwischen.

Das beschriebene Vorgehen wird mit folgendem Pseudocode formalisiert:

Algorithm 5.1: Vergleich via Sweepline (Basisvariante)

Eingabe: Textsegmente $\mathbb{S} = \{S_1, ...S_n\}$
Ausgabe: Textgemeinsamkeiten G
Textvarianten V

initialisiere G
setze $S_{n+1} = S_n$
for $i \in \{1, 2, \ldots, n\}$ **do**
    bestimme Edit-Script ES_i zwischen S_i und S_{i+1}
end
$\forall i \in [1, n]$ setze $sweepline_i$ auf Anfang von ES_i
while $\exists i$ *mit* $sweepline_i$ *zeigt nicht auf Ende von* ES_i **do**
    if $\exists i$ *mit* $ES_i(sweepline_i).op \neq$:copy **then**
        `// Anfang einer neuen Textvariante erreicht`
        vermerke Referenzen vor der gefundenen Textvariante in G
        `// Durchlaufen der Textvariante`
        while $\exists j$ *mit* $ES_j(sweepline_j).op \neq$:copy **do**
            setze $sweepline_j$ auf nächstes *:copy* in ES_j
            `// Sweepline für vorherige Segmente`
            for $k \in [j, 2]$ **do**
                setze $offset$ auf $ES_k(sweepline_k).pos\_a$
                setze $sweepline_{k-1}$ auf letzten Eintrag in ES_{k-1} mit $ES_{k-1}(sweepline_{k-1}).pos\_b = offset$
            end
            `// Sweepline für nachfolgende Segmente`
            for $k \in [j, n]$ **do**
                setze $offset$ auf $ES_k(sweepline_k).pos\_b$
                setze $sweepline_{k+1}$ auf letzten Eintrag in ES_{k+1} mit $ES_{k+1}(sweepline_{k+1}).pos\_a = offset$
            end
        end
        `// Anfang einer neuen Textgemeinsamkeit erreicht`
        vermerke Referenzen nach der abgearbeiteten Textvariante in G
    else
        $\forall i \in [1, n]$ schiebe $sweepline_i$ auf nächste Position
    end
    generiere V mittels G
    return G und V
end

Im Folgenden werden einige Codezeilen von Algorithmus 5.1 näher erläutert:
In Zeile 1 wird G mit Referenzen auf die ersten Token der Segmente initialisiert, wenn alle Edit-Scripte mit einem *:copy* beginnen. Andernfalls liegt direkt eine Textvariante vor und G wird stattdessen mit einem Tupel $(^{\aleph}{}_{\upsilon_{\mathrm{L}}})^n$ initialisiert.
In Zeile 2 wird ein zusätzliches Segment ergänzt. Hintergrund ist die Erstellung eines zusätzlichen Edit-Scripts ES_{n+1} in Zeile 4, welches das letzte Segment S_n mit sich selbst vergleicht und folglich nur aus *:copy*-Einträgen besteht. Damit wird die Dimension der Sweepline auf die Anzahl der Textsegmente verbreitert (n). Dank dieser Modifikation kann das letzte Textsegment analog zu den anderen verarbeitet werden, was zu einem Wegfall einiger Sonderfällen und damit zu einer vereinfachten Implementierung führt.
Die Erzeugung der Edit-Scripte in Zeile 4 wird in der tatsächlichen Implementierung aus Effizienzgründen statt auf den Zeichenketten der Token auf eindeutigen Zahlenwerten durchgeführt, die initial für das gesamte Vokabular von $\mathbb{S}$ vergeben werden.
In Zeile 10 müssen beim Abschließen der Textgemeinsamkeit vor der neu gefundenen Textvariante Sonderfälle im Zusammenhang mit dem Segmentende berücksichtigt und mit $^{\aleph}{}_{\upsilon_{\mathrm{L}}}$ zu gekennzeichnet werden.
In den Zeilen 13, 17 und 22 muss der Sonderfall behandelt werden, dass das Ende eines Edit-Scripts erreicht wurde. In diesem Fall zieht sich die Textvariante bis zum Ende aller Textsegmente. Entsprechend kann der Algorithmus an dieser Stelle abbrechen und G dabei ein abschließendes Tupel $(^{\aleph}{}_{\upsilon_{\mathrm{L}}})^n$ hinzufügen.
In den Zeilen 17 und 22 wird die Sweepline auf den *letztmöglichen* Eintrag verschoben. Dies stellt sicher, dass die Textvariante komplett abgearbeitet wurde. Die Notwendigkeit ergibt sich daraus, dass bei *:add*- und *:del*-Operationen das entsprechende Token nur in einem der beiden Segmente vorkommt. Im Edit-Script wird dann die vorherige Positionsreferenz wiederholt und kommt entsprechend mehrfach vor.
In Zeile 30 werden die Textvarianten V aus den in G gespeicherten Referenzen berechnet. Dabei wird eine Textvariante aus zwei aufeinanderfolgenden Tupel in G generiert, indem die gespeicherten Endreferenzen des ersten und die Startreferenzen des zweiten Tupels betrachtet werden. Pro Textsegment bildet die dazwischenliegende Textpassage die im Segment zu markierende Textvariante. Hierbei müssen diverse Sonderfälle im Zusammenhang mit leeren Textpassagen beachtet werden.

Das zurückgelieferte Ergebnis ermöglicht es nun in LERAs Ansicht für Nutzer:innen, die Textvarianten sowie die korrespondierenden Textstellen in allen Textfassungen (grau) zu markieren und in einem Variantenapparat zu listen. Die übergreifenden Textgemeinsamkeiten bleiben in diesem Standardmodus unmarkiert.

Laufzeitanalyse des Sweepline-Algorithmus

Im Folgenden wird die asymptotische Laufzeit der Heuristik untersucht.
An die Heuristik werden n Textsegmente $\mathbb{S} = \{S_1, ...S_n\}$ mit jeweils m_i Token übergeben. Sei m dabei die Anzahl der Token im längsten Segment, das heißt $m = max(m_i : i \in 1, \ldots, n])$.
Zunächst werden in Zeile 4 mittels des Algorithmus von Ukkonen n Edit-Scripte berechnet. Dieser besitzt eine asymptotische Laufzeit in $O((s+1) \cdot min(m_i, m_j))$, wobei s die Levenshtein-Distanz zwischen S_i und S_j ist [146]. Hierbei gilt $s \leq max(m_i, m_j) \leq m$, da für den ungünstigsten Fall (beide Segmente komplett unterschiedlich) alle, das heißt $max(m_i, m_j)$, Token ersetzt werden müssen.
Eine zu betrachtende Größe ist die mögliche Länge der Edit-Scripte. Dabei kann nicht einfach m angenommen werden, da das Vorkommen von *:add*- und *:del*-Operationen die Länge eines Edit-Scripts über die Länge der Textsegmente hinaus erhöhen kann: Das Edit-Script für die Zeichenketten „xy“ und „yz“ wäre eine Abfolge von je einer *:del*-, *:copy*- und *:add*-Operation. Diesem Trivialbeispiel folgend, müssten für möglichst lange Edit-Scripte in Abhängigkeit von m gleich viele *:add*- und *:del*-Operationen auftreten, die durch eine *:copy*-Operation voneinander isoliert sind, da sonst im Edit-Script eine (kostengünstigere) einzelne *:sub*-Operation stehen würde. Für längere Zeichenketten nimmt allerdings die relative Länge des Edit-Scripts bezogen auf m ab, sodass das maximale Verhältnis mit dem Faktor $1,5$ bereits beim Trivialbeispiel auftritt. Die Länge von Edit-Scripten liegt damit in $O(1,5 \cdot m)$.
Ab Zeile 7 wird die Sweepline in einer while-Schleife bewegt. Entweder wird die Sweepline in der äußeren Schleife simultan in allen Edit-Scripten eine Position nach vorne geschoben (in $O(n)$) oder die innere Schleife ab Zeile 12 wird abgearbeitet. Dabei schiebt die innere Schleife die Sweepline individuell in jedem Edit-Script voran, bis die aktuelle Textvariante abgearbeitet wurde. In beiden Fällen wird die Sweepline stets nach „vorne“ geschoben, sodass jede Position in allen Edit-Scripten genau einmal durchlaufen wird. So liegt die asymptotische Laufzeit für die Schleifendurchläufe in $O(n \cdot 1,5 \cdot m)$.
Zuletzt wird in Zeile 30 das Ausgabeformat für LERA erzeugt. Die asymptotische

Laufzeit dieses Schrittes hängt von der Anzahl der gefundenen Textgemeinsamkeiten ab. Diese liegt bei maximal $(1 + m/2)$ und tritt genau dann auf, wenn alternierend Textgemeinsamkeiten und -varianten von je einem Token zwischen den Textsegmenten identifiziert wurden. Da darauf aufbauend die Positionsreferenzen für jede der n Textfassungen berechnet wird, liegt die Gesamtlaufzeit dieses Schritts in $O(n \cdot m)$.
Die asymptotische Laufzeit für den gesamten Sweepline-Algorithmus liegt damit im ungünstigstem Fall in $O(n \cdot m^2)$ und tritt bei komplett unterschiedlichen Textsegmenten mit einer paarweisen Levenshtein-Distanz von m auf, wobei die Berechnung der Edit-Scripte maßgeblich ist. Im günstigsten Fall – komplett identische Textsegmente mit einer paarweisen Levenshtein-Distanz von 0 – liegt sie in $O(n \cdot m)$.
Für alle alignierten Segmente der vier betrachteten Textfassungen des sechsten Buchs der *Histoire des deux Indes* benötigt der vorgestellte Sweepline-Algorithmus unter drei Sekunden[8].

Erweiterungen des Sweepline-Algorithmus

Für den praktischen Einsatz in LERA gibt es wesentliche Erweiterungen der detaillierten Differenzanalyse mittels des beschriebenen Sweepline-Algorithmus.

Dazu gehört die Realisierung des in Abschnitt 4.2.2 bereits beschriebenen Filtersystems, mit dem bestimmte Textvarianten optional durch die Nutzer:innen beim Vergleich ignoriert werden können. Die Filterung der Textsegmente findet unmittelbar vor Anwendung des Sweepline-Algorithmus statt. Das gilt sowohl für Filter, die Variationen in den Token (Diakritika) oder ganze Tokenklassen (Interpunktionen) entfernen, als auch für Filter, die Token modifizieren, wie die Abbildung von Wörtern auf ihre Grundform (Lemmatisierung) oder die Normalisierung von französischen Verben im Imparfait. Hierbei kann es zum Entfernen des kompletten Inhalts einzelner Token kommen, beispielsweise wenn mittels eines Vergleichsfilters Seitennummern ignoriert werden sollen und diese entsprechend aus den Zeichenketten gelöscht werden. Die resultierenden Leerstellen werden bei der Filterung erkannt, mittels einer Datenstruktur gespeichert und die Tokens schließlich vom Ver-

[8] Die Laufzeit betrug für zehn Messungen im Mittel 2,98 Sekunden bei einer Standardabweichung von 0,21 Sekunden. Getestet wurde dabei auf einem Laptop mit 64-bit-Betriebssystem (Ubuntu 22.04), vier Kernen mit einer Taktung von je 2,40GHz (Intel® Core™ i7-5500U) und acht GiB DDR3-Arbeitsspeicher mit einer Geschwindigkeit von 1600 MT/s.

gleich ausgenommen. Mittels der gespeicherten Informationen werden sie nach der Abarbeitung der Sweepline wieder in die berechneten Textgemeinsamkeiten und -varianten eingefügt, sodass die Texte vollständig in der Nutzeransicht dargestellt werden können.

Ein besonderer Filter stellt der 80-%-Filter dar. Dieser ignoriert Textvarianten, wenn mindestens 80 % der in den Token vorkommenden Zeichen übereinstimmen. Hintergrund ist der Vergleich von Sprachen in denen hauptsächlich Wortwurzeln statt der vollständigen Wörter geschrieben werden, das heißt, die Vokale werden nur mitgedacht statt ausgeschrieben, so wie in den arabischen Textfassungen von *Kalīla and Dimna*. Die gewünschte Anwendung des 80-%-Filters muss dazu direkt bei der Berechnung der Edit-Scripte berücksichtigt werden. Hierzu wurde der in LERA verwendete Algorithmus von Ukkonen, so modifiziert, dass er statt eines exakten Vergleichs von zwei Token auch einen unscharfen Vergleich zulässt. Dieser findet direkt auf den Zeichenketten der Token und nicht auf den sie sonst repräsentierenden Zahlenwerten statt. Die resultierenden Edit-Scripte können anschließend wie bisher vom Sweepline-Algorithmus verarbeitet werden.

Der Sweepline-Algorithmus liefert die grundlegenden Daten zum Einfärben der Textvarianten in der Nutzeroberfläche, wie bereits in Abschnitt 4.2.5 beschrieben.

Für den Grundmodus, welcher die gefundenen Textvarianten samt der korrespondierenden Textpassagen in den anderen Textfassungen grau hervorhebt, können die in der zurückgelieferten Datenstruktur für Textvarianten (V) aufgeführten Referenzen auf die entsprechenden Token direkt genutzt werden.

Ein weiterer Farbmodus erlaubt das Hinzuschalten der farblich codierten Operationen aus den Edit-Scripten, welche dafür durch eine Erweiterung des Sweepline-Algorithmus aufbereitet und zurückgeliefert werden. Diese optionalen Farbmarkierungen sind im Kontrast zu LERAs gleichrangigem Vergleichsprinzip allerdings abhängig von der gewählten Reihenfolge der Textfassungen.

Für die Farbmodi zum Einfärben von Einzelstellen, Paaren und Gruppen innerhalb von Textvarianten wird hingegen eine zusätzliche wortweise Alignierung für jede Textvariante ausgeführt und zurückgeliefert. Diese Erweiterung des Sweepline-Algorithmus liefert die notwendige Datengrundlage sowohl für die drei Farbmodi als auch für die Partitursynopse, einer alternativen Darstellung zur Spaltensynopse.

Der Sweepline-Algorithmus wird auch zur Bestimmung der Textvarianten zwischen

transponierten Textsegmenten genutzt, die während der Alignierung erkannt wurden. Dazu wird der Algorithmus ein zweites Mal für die um die Kopien der transponierten Segmente erweiterten Folge $\mathbb{S}$ aufgerufen, insofern solche vorliegen. Durch die Vorberechnung beider Versionen kann die Anzeige kopierter Segmente reaktionschnell über die Nutzeroberfläche umgeschaltet werden.

5.4.4 Weiterführende Bemerkungen

Zusätzlich zur detaillierten Differenzanalyse auf Tokenebene und der Alignierung auf Segmentebene wird LERA aktuell um eine dritte Vergleichsstufe erweitert, die auf der Fassungsebene angesiedelt ist. Wie im nachfolgenden Kapitel in Abschnitt 6.4 diskutiert, dient diese dazu, umfangreiche Korpora mit sehr vielen Textfassungen zu analysieren. Ziel ist dabei die Auswahl einer vielversprechenden Menge von Textfassungen, bevor die eigentliche Kollationierung auf Segment- und Tokenebene folgt. Der bestehende Prototyp wird im Rahmen dieser Dissertation allerdings ausschließlich in Hinblick auf die bereits fortgeschrittenen, visuellen Analysemöglichkeiten im folgenden Kapitel diskutiert. Die algorithmischen Grundlagen, die eine Echtzeitberechnung ermöglichen, sind aktueller Forschungsgegenstand und werden daher an dieser Stelle ausgespart.

Kapitel 6

Exploration von Textvarianten

In diesem Kapitel werden die ergänzenden visuellen Komponenten von LERA zur Navigation und Analyse von großen Textmengen diskutiert.
Die grundlegende Form der Darstellung in LERA ist die synoptische Gegenüberstellung alignierter Textsegmente samt der ausgezeichneten Varianten. Bei diesem Close-Reading-Ansatz werden die Textsegmente entweder als Spalten- oder Partitursynopse im Volltext mit ergänzenden Variantenapparaten repräsentiert. Diese Form der Darstellung ist zur Analyse von Einzelstellen, das heißt einzelner Textvarianten im Detail, ausgelegt. Demgegenüber sind die untersuchten Textfassungen vieler Editionsprojekte sehr umfangreich. In diesem Vergleichsszenario sind Überblickdarstellungen sehr hilfreich, welche die vielen einzelnen Textvarianten in aggregierter Form repräsentieren und so den Nutzer:innen den Einstieg in die Analyse erleichtern. Gleichzeitig können diese Visualisierungen wiederum Funktionen zum Entdecken interessanter Einzelstellen bieten, die neben der intendierten Suche auch unerwartete Funde ermöglichen. Ein dritter wichtiger Aspekt sind Hilfestellungen zur Orientierung und Navigation in großen Textmengen. LERA bietet mit einer Suchfunktion, der Navigations- und Übersichtsleiste CATview und stabilen Wortwolken drei Komponenten, welche die Arbeit mit sehr umfangreichen Textfassungen erleichtern. Insbesondere die Kombination dieser in Teilen interaktiven Distant-Reading-Ansätze bietet einen weiteren Zugang zu den Textvarianten. Dieses Thema wurde bereits im Rahmen eines Konferenzbeitrags von 2016 diskutiert, siehe [139], aus dem die wesentlichen Aspekte und einige Formulierungen in die folgenden Ausführungen übernommen wurden.
Zusätzlich zur Handhabung einiger umfangreicher Textfassungen im direkten Vergleich ist auch der generelle Umgang mit einer großen Anzahl von Textfassungen

für manche Editionsvorhaben relevant. LERA bietet mit der Korpussynopse, die in Abschnitt 6.4 betrachtet wird, hierfür eine zusätzliche, interaktive Komponente zur visuellen Analyse. Diese stellt die Textfassungen des gesamten Korpus in ihrer Gänze in Bezug und fungiert so als eine weitere, vorgelagerte Vergleichsstufe. Ziel dabei ist es, einen Überblick über das gesamte Korpus zu erhalten und so interessante Teilmengen von Textfassungen zu identifizieren, die dann als Synopse mit den bisherigen Methoden verglichen werden können.

6.1 Suchfunktion

LERAs Suchfunktion ermöglicht es, innerhalb der synoptischen Gegenüberstellung von Textfassungen nach Wörtern und Phrasen zu suchen. Dazu wurde eine Suchmaske in der Titelleiste integriert, die nach einer Suche um die Anzahl der Suchtreffer sowie zwei Pfeiltasten zum Anspringen der entsprechenden Stellen in der Synopse erweitert wird. Die Suchtreffer selbst werden farbig (gelb) im Volltext hervorgehoben, wobei der aktuell angesprungene Treffer noch eine besondere Hervorhebung erfährt. Diese Hervorhebungen überdecken die farblichen Markierungen von Textvarianten, falls nötig. Abbildung 6.1 zeigt die Oberfläche für eine beispielhafte Suche.

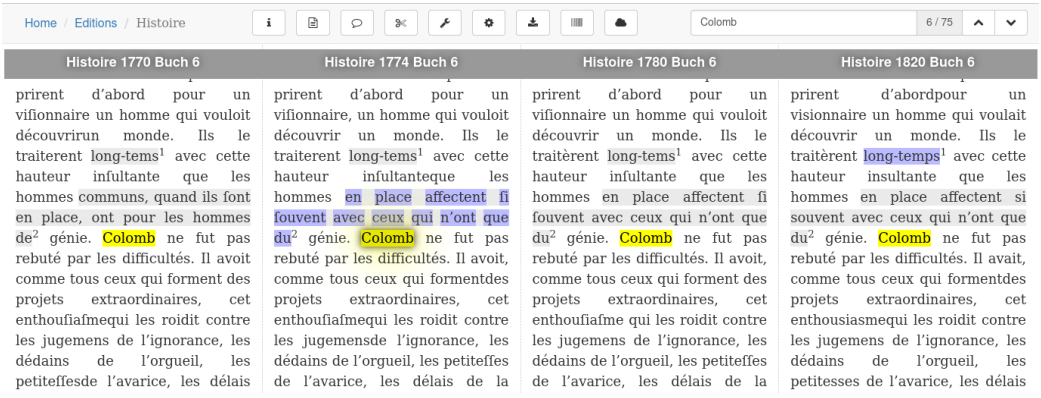


Abbildung 6.1: LERAs Suchfunktion am Beispiel: Das Schlagwort „Colomb“ wurde in einer Synopse der *Histoire des deux Indes* mittels einer integrierten Schaltfläche (oben rechts im Bild) gesucht und die 75 Vorkommen im Volltext farblich (gelb) markiert. Mittels der Pfeiltasten wurde das sechste Vorkommen angesprungen, welches nochmals besonders hervorgehoben wird.

Die Suchtreffer werden im Backend von LERA berechnet. Dies hat gegenüber der Frontend-seitigen Berechnung zwar den Nachteil des notwendigen Datenaustauschs zwischen Server und Client, erlaubt aber zusätzliche Transformationen zur Normalisierung der Daten sowie die Nutzung vorberechneter Indizes zur Steigerung der Effizienz. Technisch wurde dies mithilfe von Thinking Sphinx [7] realisiert, welche die Volltextsuchmaschine Sphinx für die in Ruby-on-Rails als ActiveRecords verwalteten Daten bereit stellt und im Falle von LERA auf den Token arbeitet. Zur Normalisierung werden beispielsweise Großbuchstaben durch Kleinbuchstaben ersetzt und kombinierende Unicodes, wie Diakritika, entfernt. Für die Token ist dazu in der Datenbank neben ihrer den Nutzer:innen darzustellenden Form (in NFD) auch eine normalisierte Variante als *Normalization Form Compatibility Composition* (NFKC) nach Unicode-Standard[154] gespeichert. Der Kompatibilitätsmodus (Compatibility) schafft dabei eine Äquivalenz zwischen unterschiedlichen Unicodes, die dasselbe abstrakte Zeichen darstellen. Dadurch werden unter anderem Ligaturen aufgetrennt (fi → fi) oder auch Schriftartvarianten eines Buchstaben standardisiert, wie beispielsweise für „H“, „ℌ“ und „ℍ“. Die beschriebene Normalisierung wird gleichsam auf die Suchphrase angewendet und erlaubt so eine gewisse Unschärfe bei der Suche.

Die Suchfunktion in LERA ermöglicht eine Sternsuche, die auf den entsprechenden, in Thinking Sphinx eingebauten Mechanismus zurückgreift. So können Nutzer:innen das Sternchen (*) in der zu suchenden Phrase platzieren, um eigenständig eine Unschärfe zu realisieren. Die Suchmaschine liefert dann Suchtreffer, bei denen die Sternchen durch beliebige Zeichenketten aufgefüllt sind. Sternchen können dabei auch an beliebigen Positionen der Suchphrase platziert werden, um Pre-, In-, oder Suffixe generieren zu lassen. Es besteht allerdings die Einschränkung, dass in der Suchphrase jeweils drei zusammenhängende Zeichen vorgegeben werden müssen, um den im Backend aufgebauten Index nicht zu groß sowie die Anzeige der Suchtreffer nicht zu unübersichtlich werden zu lassen.

Aufbauend auf der Suche nach einzelnen Termen, besteht in LERA auch die Möglichkeit zur Suche nach Phrasen, die sich aus mehreren Wörtern zusammensetzen. Dazu wird die Suchphrase im Backend zerlegt und für die einzelnen Terme eine Suche durchgeführt. Die Ergebnisse werden anschließend kombiniert und für die Darstellung im Frontend aufbereitet. Über eine Einstellmöglichkeit in der Nutzeroberfläche können dabei auch Phrasen mit Abweichungen in der Wortreihenfolge, Lücken oder Teilphrasen als Suchtreffer erlaubt werden.

Um die im Backend berechneten Suchtreffer schließlich in der HTML-Ansicht der

Synopse verfügbar zu machen, werden sie samt Positionsangaben ans Frontend weitergeleitet. Dort übernimmt eine dafür für LERA geschaffene Javascript-Komponente die Verwaltung und Darstellung. Für jeden Suchtreffer extrahiert sie mittels der Positionsangaben das HTML-Fragment des Segments, welches den Suchtreffer enthält, und durchsucht die Konkatenation der darin enthaltenden Textknoten – bereits als Suchtreffer markierte Fragmente werden dabei ausgenommen – mithilfe der in allen gängigen Browsern unterstützten *search*-Funktion von Javascript[1]. Um sicherzustellen, dass dabei der richtige Suchtreffer gefunden und markiert wird, werden die Suchtreffer in der Reihenfolge ihres Auftretens im Text abgearbeitet und mit den selben Anzeigefiltern normalisiert, wie sie auch für das Rendern der HTML-Darstellung des Segments genutzt wurden. Zudem kommt ein regulärer Ausdruck zum Einsatz, mit dem Wortgrenzen gewahrt bleiben, sodass nicht fälschlicherweise übereinstimmende Substrings in anderen Wörtern statt der eigentlichen Suchtreffer gewählt und markiert werden. Die Markierung selbst geschieht über das Umschließen des ermittelten Suchtreffers mit einem <mark>-Tag, das zudem ein Positionsattribut erhält und so das direkte Anspringen der Suchtreffers ermöglicht.

6.2 CATview

CATview, ein Akronym für *the Colored & Aligned Texts view*, ist eine interaktive Visualisierung, welche zur übersichtlichen Darstellung der Ergebnisse eines Textvergleichs konzipiert wurde und gleichzeitig die Navigation darin vereinfachen soll. Sie wurde im Rahmen der Arbeiten für LERA mitentwickelt, ist aber auch als eigenständiges Tool verfügbar und wurde beispielsweise in LAKomp[2] integriert. Im Folgenden werden die Funktionalitäten, ihre technische Umsetzung samt der aufgetretenen Probleme, die Einbindung in LERA sowie die Verknüpfung mit der Suchfunktion vorgestellt. Als Grundlage hierfür diente ein Konferenzartikel von 2015 ([117]).

[1] Die Funktion *search* wurde bereits im ECMAScript 1 Standard von 1997 spezifiziert.

[2] LAKomp ist ein Tool zur Lemmatisierung, Annotation und Komparation frühneuhochdeutscher Texte und wurde wie LERA im Rahmen des Projekts *Semi-automatische Differenzanalyse von komplexen Textvarianten (SaDA)* entwickelt. Eine Demo ist unter `https://lakomp.uzi.uni-halle.de` (besucht am 11.12.2023) Verfügbar.

6.2.1 Funktionalitäten

Grundlegend ist CATview eine tabellarische Darstellung alignierter Segmente verschiedener Textfassungen. In der horizontalen Standardausrichtung gibt es dabei jeweils eine Zeile pro Textfassung, deren Segmente wiederum fortlaufend und abstrahiert als Rechtecke in den Zellen der Tabelle visualisiert werden. Wurden die Segmente verschiedener Textfassungen aligniert, werden sie in der gleichen Spalte platziert. Die Spalten werden angelehnt an den Alignierungsalgorithmus von LERA im Folgenden als Alignierungskanten (engl. edges) bezeichnet. CATview fällt mit der beschriebenen Grundfunktion in die Gruppe der *Sequence-aligned Heat Maps* laut [156]. Abbildung 6.2 zeigt CATview für einen Auszug der *Histoire des deux Indes*.

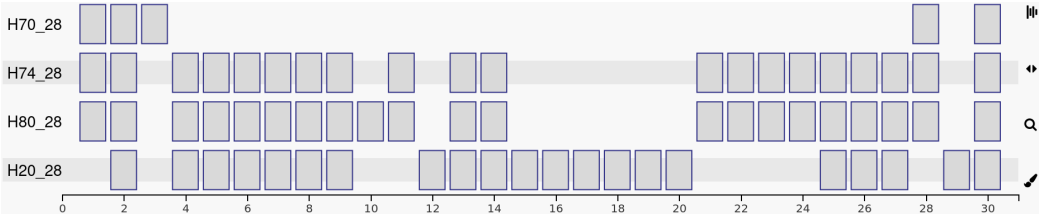


Abbildung 6.2: CATviews Grundfunktion zur Darstellung einer Alignierung am Beispiel eines Auszug der *Histoire des deux Indes*: Mittels der grauen Rechtecke werden die Segmente der vier Textfassungen zeilenweise dargestellt. Wurden Segmente aus unterschiedlichen Fassungen aligniert, werden sie dabei in der selben Spalte platziert.

Um CATview für verschiedene Einsatzszenarien und Lesegewohnheiten flexibel zu halten, wurde ihre Ausrichtung variabel gestaltet. So lässt sie sich per Option leicht an der oberen oder unteren sowie der linken oder rechten Kante des Bildschirms platzieren. Zudem können beide Achsen, das heißt die Reihenfolge der Textfassungen oder Alignierungskanten, individuell gespiegelt werden, um beispielsweise auch den Lesegewohnheiten in Sprachen mit der Schreibrichtung von Rechts-nach-Links, wie dem Hebräischen oder Arabischen, gerecht zu werden.
Zur vereinfachten Handhabung sehr großer Mengen alignierter Segmente wurde eine Zoomfunktion in CATview integriert, die mit dem Mausrad bedient wird. Der im gezoomten Zustand dargestellte Bereich kann durch das Drücken und Halten der linken Maustaste verschoben werden.

CATview wurde sowohl zur Übersicht als auch als Hilfsmittel zur Navigation konzipiert. So ist die Visualisierung mit der synoptischen Volltextdarstellung der Alignierung von LERA verknüpft. Zum einen zeigt ein orangefarbener Balken in CATview die aktuelle Scrollposition (225) in der Synopse an. Zum anderen scrollt die synoptische Darstellung zum entsprechenden Textsegment, sobald ein Rechteck angeklickt wird. CATview zeigt zudem den Segmentbezeichner an, sobald die Maus über ein Rechteck der entsprechenden Alignierungskante in CATview hovert. Abbildung 6.3 veranschaulicht einige der genannten Navigationshilfen am Beispiel.

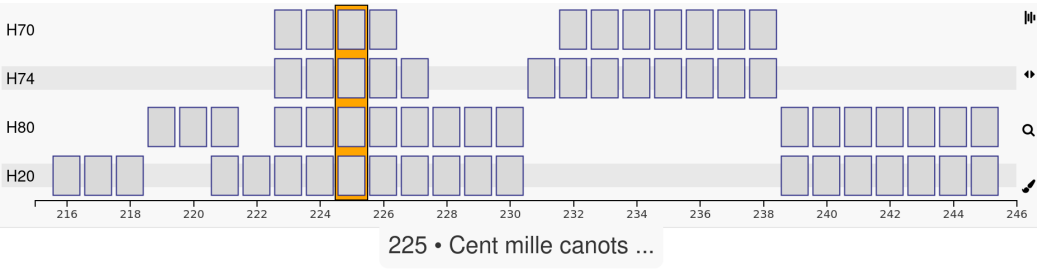


Abbildung 6.3: CATview kann als Navigationshilfe für umfangreiche Textfassungen genutzt werden, hier am Beispiel des sechsten Buchs der *Histoire des deux Indes*. Mittels der integrierten Zoomfunktion wurden die alignierten Textsegmente 216 bis 245 fokussiert. Der orangefarbene Balken kann mit einer Volltextdarstellung verknüpft werden und zeigt die aktuelle Scrollposition darin an. Am unteren Rand ist zudem die Segmentnummer und eine kurze Vorschau zu sehen, welche in Form individueller Segmentbezeichner in CATview spezifiziert werden kann.

Die Rechtecke in CATview können zur Visualisierung weiterer Informationen des Textvergleichs genutzt werden.
Zum Ersten wurde eine Funktionalität zur individuellen Anpassung der Höhe implementiert. So lässt sich über die Höhe eines Rechtecks die Textmenge des dazugehörigen Textsegments andeuten.
Zum Zweiten ist die Anpassung der Färbung nützlich. In LERA werden so zum einen die Varianzen der Segmente innerhalb einer Alignierungskante, das heißt die Ergebnisse des detaillierten Textvergleichs auf Tokenebene, visualisiert. Dabei bestimmt der Anteil von Token in Textvarianten bezüglich der gesamten Token des Segments einen Wert im Intervall $[0, 1]$, welcher auf eine blaue Farbskala übertragen wird. Der Farbton ist umso dunkler, je größer der Anteil an Textvarianten ist. Liegt der Wert bei 0, das heißt, es sind keine Textvarianten im Segment enthal-

ten, wird für das entsprechende Rechteck das Standard-Grau genutzt, um so einen besonderen Kontrast für identische Textstellen zu schaffen. So lassen sich Stellen mit Textvarianten sowie ihr Grad an Varianz bereits in der Übersichtsleiste visuell erkennen. Die Färbung wird zum anderen für die Hervorhebung von Suchtreffern genutzt, indem wahlweise die einzelnen Rechtecke oder die ganze Spalte gelb markiert werden. Diese Funktion hilft, die Verteilung eines Begriffs oder Themas in den Textfassungen einzuschätzen.
Zum Dritten können in CATview zusätzliche Rechtecke platziert werden, die nur aus einer gestrichelten Umrandung bestehen und sich so von den regulären Rechtecken abheben. Sie werden in LERA zur ergänzenden Darstellung transponierter Segmente genutzt. CATview erlaubt dabei das Verknüpfen von Rechtecken in der Gestalt, dass durch das Hovern über ein Rechteck andere farblich hervorgehoben werden können. So werden im Fall von Transpositionen die originalen und kopierten Segmente visuell miteinander verknüpft, um den Nutzer:innen eine leichtere Zuordnung zu ermöglichen. Abbildung 6.4 veranschaulicht die genannten Modifikation der Rechtecke in CATview am Beispiel.

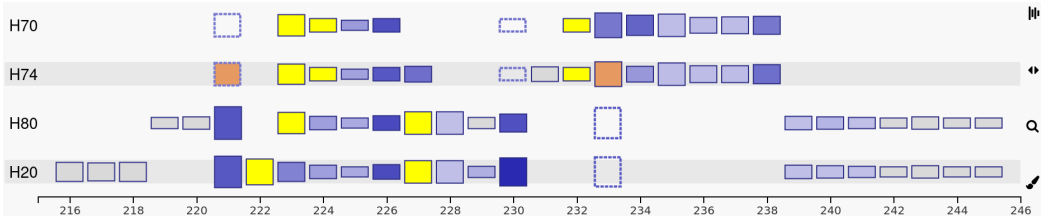


Abbildung 6.4: Visualisierung weiterer Informationen mittels der Rechtecke in CATview. Die Segmenthöhe veranschaulicht den Textumfang der repräsentierten Segmente, die Blautöne die Textvarianz. Für Suchtreffer wird die Füllfarbe mit Gelb ersetzt. Zudem werden durch zusätzliche, gestrichelte Rechtecke vertauschte Segmente angezeigt, die nicht direkt aligniert werden konnten. Dabei kann mittels der Maus die Position des originalen Segments in Orange hervorgehoben werden, wie im Beispiel anhand des Segments (221) und seiner Kopie (233) in Textfassung H74 zu sehen.

In CATview wurden weitere interaktive Funktionen integriert. Dazu gehört das Auswahlwerkzeug, welches das Aufziehen einer Box über die Rechtecke und somit die Auswahl von Textsegmenten erlaubt. Die Enden der Auswahlbox werden nach dem Aufziehen auf diskrete Werte zwischen den Rechtecken verschoben, um die Aus-

wahl eindeutig zu halten und den Nutzer:innen auch so widerzuspiegeln. Dabei werden Randfälle abgefangen und die Zoomfunktion oder etwaige Achsenspiegelungen berücksichtigt. In LERA wird die Segmentauswahl an die stabilen Wortwolken weitergeleitet (siehe Abschnitt 6.3.2).
Eine andere interaktive Funktion von CATview erlaubt das Tauschen der Reihenfolge der Textfassungen. Wenn eingeschaltet, können die Namen der Textfassungen per Maus geklickt und verschoben werden (Drag'n'Drop), was in der Folge auch die Anordnung der dazugehörigen Zeilen ändert. Diese Funktionalität wurde primär für die Korpussynopse entwickelt (siehe Abschnitt 6.4.1). Auf LERAs synoptische Darstellung hat das Vertauschen im aktuellen Entwicklungsstand keine Auswirkung, sodass die Funktion abgeschaltet ist. Abbildung 6.5 gibt einen Eindruck von CATviews Auswahlwerkzeug sowie der interaktiven Anpassung der Zeilenanordnung.

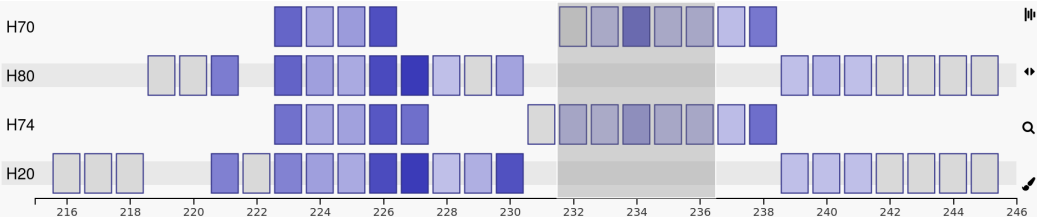


Abbildung 6.5: Anwendung interaktiver Komponenten von CATview: Mittels der Drag'n'Drop-Funktion wurde die Reihenfolge der Textfassungen H74 und H80 getauscht, mittels des Auswahlwerkzeugs ein Bereich von 232 bis 236 aufgespannt.

6.2.2 Technische Umsetzung und Integration in LERA

CATview wurde als Visualisierung für den Einsatz in Webseiten konzipiert. Im Kern handelt es sich bei der technischen Umsetzung um eine Scalable Vector Graphic (SVG), welche dynamisch mittels Javascript manipuliert wird. Dabei baut der Quellcode von CATview auf Methoden der Bibliothek d3.js (Data-Driven Documents) [15] auf, welche speziell für die Datenvisualisierung entwickelt wurde und das Manipulieren von SVG-Elementen vereinfacht. Auf weitere Abhängigkeiten, wie der Bibliothek jQuery, wurde für CATview verzichtet, um die Weitergabe und Wartung möglichst einfach zu halten. Neben dem Quellcode in Javascript (ca. 1.500 Zeilen) gehören zu CATview noch einige Dutzend Zeilen Cascading Style Sheets (CSS), um das Layout einiger Komponenten zu spezifizieren.

Die zu visualisierenden Daten werden CATview in Form von JSON-Objekten mitgeteilt und können initial mittels einer Server-Anfrage oder direkt via Javascript-Funktionen übertragen werden. Die eigentliche Berechnung der Daten passiert dabei serverseitig in LERA. Dazu gehören die Alignierung der Segmente, die Namen von Textfassungen, Segmentbezeichner, die sich aus dem detaillierten Textvergleich ergebenden Varianzwerte (Farben), die normierte Höhe der Rechtecke entsprechend der Textmenge, die zusätzlichen Textsegmente (Kopien) samt der Verknüpfung mit ihrer eigentlichen Position für die Visualisierung von Transpositionen sowie die Positionen von Suchtreffern.

CATview enthält zudem Methoden zur Synchronisation, die eine nachträgliche Aktualisierung der Daten in Gänze oder punktuell erlauben. So werden beispielsweise Änderungen in der Alignierung bei manueller Intervention durch Nutzer:innen in LERA direkt an CATview weitergeleitet und die Übersichtsleiste entsprechend punktuell aktualisiert (siehe Abschnitt 4.2.3). Gleiches gilt für geänderte Einstellungen für den detaillierten Textvergleich, was eine Änderung der Farbwerte der Rechtecke nach sich zieht. Die Daten für den Scrollspy in CATview werden ebenfalls durch LERA stetig aktualisiert, sobald in der synoptischen Volltext-Repräsentation gescrollt wird.

Bei der Einbindung von CATview in eine Webseite können darüber hinaus spezielle Callback-Funktion definiert werden, die nach gewissen Interaktionen mit CATview aufgerufen und ausgeführt werden. Mit diesem Vorgehen konnte der Quellcode von CATview gekapselt und dennoch eine starke Verknüpfung mit anderen Komponenten von LERA realisiert werden. Dazu gehören setzbare Callback-Funktionen für die Reaktion auf den Klick eines Rechtecks, das Selektieren von Rechtecken via Auswahlwerkzeug oder die Änderung der Reihenfolge der Textfassungen (via Drag'n'Drop).

CATview ist als eigenständige Open Source Software verfügbar und besitzt eine englischsprachige Homepage samt Demonstratoren und einer Dokumentation der Funktionalitäten, die unter `https://catview.uzi.uni-halle.de` erreichbar ist. Der Quellcode liegt in einem auf der Homepage verlinkten Repositorium und ist unter MIT-Lizenz verfügbar.

6.3 Interaktive Wortwolken

LERA bietet mit den integrierten Wortwolken eine weitere Distant-Reading-Visualisierung zur Analyse der kollationierten Textfassungen an. Wortwolken dienen zur optisch ansprechenden Darstellung ausgewählter Wörter aus einer (meist) sehr umfangreichen Textmenge. Abbildung 6.6 zeigt dies exemplarisch für diese Dissertation.

Abbildung 6.6: Eine Wortwolke mit 50 häufig in dieser Dissertation vorkommenden Begriffen. Die Darstellungsgröße steigt dabei mit Anzahl der Vorkommen.

Welche Wörter dabei visualisiert werden, weil sie als relevant betrachtet werden, ist vom Anwendungsszenario oder den Präferenzen der Nutzer:innen abhängig. Gleiches gilt für die Art der Darstellung, für die es sehr unterschiedliche Ansätze je nach Einsatzgebiet gibt [87, 140]. Eine beispielhafte Anwendung samt Umsetzung wird in [90] beschrieben. Mithilfe einer automatischen Schlagwort-Extraktion werden aus einer Nachrichtensendung die wesentlichen Begriffe und Phrasen der letzten sechs Stunden extrahiert und als dreidimensionale Wortwolke aufbereitet. Diese sind wiederum mit den Beiträgen selbst verknüpft und erlauben den Nutzer:innen so einen übersichtlichen Zugriff.

6.3.1 Wortwolken in LERA

In LERA dienen Wortwolken als ergänzendes Hilfsmittel zum Vergleich verschiedener Textfassungen. Als eine aggregierte Form des Inhalts, wobei wichtige Themen

hervorgehoben werden können, bieten sie einen leichten Einstieg in den inhaltlichen Vergleich der Textfassungen, bevor die Texte selbst im Detail betrachtet werden. Dazu wird in LERA je Textfassung eine eigene Wortwolke generiert. Diese können jederzeit oberhalb der synoptischen Gegenüberstellung der Textfassungen eingeblendet werden. Die Wortwolken werden dabei analog zu den Volltextrepräsentationen nebeneinander angeordnet, um zum einen die intuitive Zuordnung zu den Textfassungen und zum anderen den visuellen Abgleich untereinander für die Nutzer:innen zu erleichtern.

Für die Generierung der Wortwolken wurden diverse Einstellmöglichkeiten integriert, welche sowohl die Bestimmung der relevanten Wörter als auch ihre Darstellung betreffen.

Zum Ersten kann die grundlegende Auswahl der Wörter auf bestimmte Wortarten eingeschränkt werden, insofern sie beim Import der Textfassungen oder nachträglich über die Nutzeroberfläche von LERA mithilfe des TreeTaggers ausgezeichnet wurden (siehe Abschnitt 4.2.2). Zudem kann eine minimale Wortlänge festgelegt werden, was als eine einfache Heuristik zum Herausfiltern von Stoppwörtern dienen kann, wenn beispielsweise die Filterung über Wortarten nicht verfügbar oder unzureichend ist.

Zum Zweiten lässt sich das Maß zur Bewertung der Relevanz der Wörter ändern. Gemäß einer einstellbaren Anzahl werden entsprechend viele, als am besten bewertete Wörter dargestellt. Die verfügbaren Maße werden dabei individuell pro Textfassung angewendet, um die für sie relevantesten Wörter für den Vergleich zu bestimmen. Alternativ wäre eine fassungsübergreifende Bewertung denkbar, um zunächst die Menge der relevantesten Wörter des Werkes zu bestimmen, bevor in einem zweiten Schritt die fassungsbezogene Bewertung dieser Wörter für den Vergleich folgen würde. Beide Vorgehen haben ihre Berechtigung. Momentan wird allerdings nur die erste Variante unterstützt.

Als Bewertungsmaß wurde neben der absoluten Häufigkeit das TF-IDF-Maß mit verschiedenen Parametern in LERA integriert. Beim zugrunde liegenden IDF-Wert [142] handelt es sich ursprünglich um einen Ansatz, um die Relevanz von Wörtern zur Klassifizierung von Dokumenten zu bewerten. Bei der Bestimmung des darauf aufbauenden TF-IDF-Wertes wird die Häufigkeit eines Wortes innerhalb eines Dokuments – die Term Frequency (TF) – mit der inversen Häufigkeit des Wortes in der Gesamtheit der Dokumente – Inverse Document Frequency (IDF) – multipliziert.

So können Schlagwörter gefunden werden, die für ein Dokument sehr wichtig, in der Gesamtheit der Dokumente aber relativ selten sind, also vermeintlich gut zur Abgrenzung dienen. Aufgrund seiner Effektivität ist dieses Maß in Anwendungen aus dem Bereich des Information Retrieval weit verbreitet [5, 127].

Bei der Adaption für die Wortwolken in LERA wird das TF-IDF-Maß, wie oben beschrieben, individuell auf die einzelnen Textfassungen bezogen. Ein Textsegment entspricht dabei einem Dokument, die Gesamtheit aller Segmente der Textfassung der Menge von Dokumenten. Während die Bewertung mittels absoluter Häufigkeit Wörter zurückliefert, die insgesamt eine hohe Relevanz in der ganzen Textfassung besitzen, bietet sich das TF-IDF-Maß an, um thematisch relevante Wörter zu finden, die nur an wenigen Stellen vorkommen. Während die Berechnungsweise der Inversen Dokumenthäufigkeit (IDF) in LERA vordefiniert ist, kann für die Worthäufigkeit (TF) aus verschiedenen Optionen gewählt werden. Dazu gehört zum einen die Art der Normalisierung bezogen auf die Worthäufigkeit innerhalb eines Textsegments, wobei optional mit dem häufigsten Wort im Segment und/oder der Segmentlänge normalisiert werden kann. Und zum anderen wie die Aggregation der Worthäufigkeit über die Segmente – mit entsprechendem Vorkommen des Wortes – hinweg vollzogen werden soll, wobei das Maximum oder der Durchschnitt als Optionen zur Auswahl stehen.

Neben Filterung und Bewertung der Wörter kann zum Dritten die Darstellungsform der Wortwolken in LERA gewählt werden. Zur Auswahl stehen einfache Listen, die für Wortwolken klassisch elliptische Form sowie animierte, drei-dimensionale Kugeln, die mittels der Javascript-Bibliothek TagCanvas [21] realisiert wurden. Zu den wählbaren Darstellungsformen gehören auch die stabilen Wortwolken, die im nächsten Abschnitt näher diskutiert werden.

Wie auch die Suchfunktion und CATview stellt die technische Umsetzung der Wortwolken in LERA ein Zusammenspiel aus Backend und Frontend dar. Die grundlegenden Daten werden initial beim Öffnen der Wortwolken im Backend berechnet und via JSON-Objekt an den Client geschickt. Dort sorgt eine Javascript-Komponente für die Verarbeitung und Darstellung der Daten. Sie reagiert zudem auf Änderungen der Optionen im dafür vorgesehenen Menü, wobei ein Großteil der Neuberechnung direkt im Frontend geschieht. Darüber hinaus haben Änderungen der Alignierung durch Nutzereingriff Auswirkungen auf die Wortwolken. In diesem Fall wird der Datenbestand entsprechend automatisch aktualisiert.

Abbildung 6.7 zeigt LERAs Wortwolken und die dazugehörige Nutzeroberfläche am Beispiel.

Abbildung 6.7: LERAs interaktive Wortwolken am Beispiel des sechsten Buchs der *Histoire des deux Indes* in vier Textfassungen. Mittels des Optionsmenüs wurden die 20 häufigsten Substantive mit einer Mindestlänge von sechs Buchstaben gewählt und als zwei-dimensionale, elliptische Wortwolken visualisiert.

6.3.2 Verknüpfung mit der Suchfunktion und CATview

Die Ergebnisse der Suche werden in den interaktiven Wortwolken ebenfalls farbig hervorgehoben, was das Auffinden der Schlagworte und damit den Vergleich der Relevanz beziehungsweise ihrer Häufigkeit in den verglichenen Textfassungen erleichtert. Zudem löst wiederum ein Klick auf einen der angezeigten Begriffe in den Wortwolken die Suche aus, sodass interessanten Funden in den Wortwolken direkt nachgegangen werden kann.

Auch CATview ist mit den Wortwolken gekoppelt. Das Auswahlwerkzeug der Übersichtsleiste zum Festlegen eines Textbereichs hat direkten Einfluss auf die Datengrundlage der Wortwolken, indem nur noch die ausgewählten Segmente für die Bestimmung der relevanten Begriffe betrachtet werden. Die Zusammensetzung der Wortwolken wird dabei umgehend für jede neue Auswahl aktualisiert. Über diese Kopplung lässt sich ein Textbereich für die genauere Analyse mit den Wortwolken festlegen, was sich beispielsweise anbietet, wenn ein Abschnitt des Textes wegen der in CATview dargestellten Textvarianz oder Verteilung von Suchbegriffen besonders interessant erscheint. Abbildung 6.8 zeigt ein solches, aus [139] entnommenes, Beispiel.

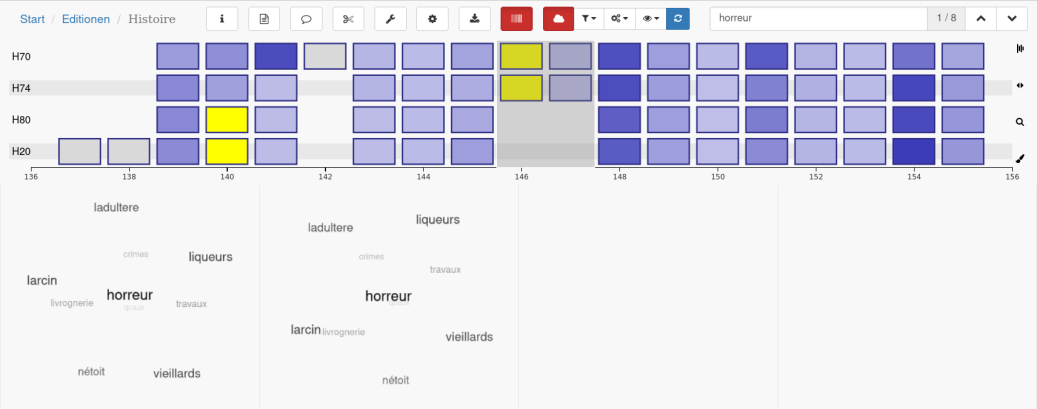


Abbildung 6.8: Die Verknüpfung der Wortwolken mit der Suchfunktion und CATview am Beispiel: In CATview wurde der interessant erscheinende Abschnitt 146 bis 147, welcher nur Textsegmente in den beiden Textfassungen H70 und H74 aufweist, mittels des Auswahlwerkzeugs markiert. Die Wortwolken wurden darauf hin automatisch vom System aktualisiert. Sie zeigen nun eine Übersicht relevanter Wörter dieser, aus den späteren Textfassungen gestrichenen Textpassage. Ein Klick auf den Begriff „horreur“ in den Wortwolken löst eine Suche aus, die wiederum die Suchtreffer in CATview gelb markiert und so einen Hinweis dafür liefert, dass sich Teile der Passage in den späteren Textfassungen in Abschnitt 140 wiederfinden.

Die Auswahlbox kann mittels Drag’n’Drop in der Übersichtsleiste bewegt werden. Durch diese Bewegung verändern sich synchron die Wortwolken, sodass stets die Themen der aktuell gewählten Textpassage angezeigt werden. Dadurch erzeugt die Bewegung eine Art interaktive Animation, anhand derer die inhaltliche Entwicklung der Texte veranschaulicht wird – neue Themen tauchen auf, nehmen an Bedeutung zu und wieder ab, bis sie verschwinden und gegebenenfalls an anderer Stelle im Text wieder auftauchen.

Um das beschriebene Zusammenspiel der interaktiven Visualisierungen zu verbessern, beziehungsweise für Nutzer:innen deutlicher effektiver zu gestalten, wurden *stabile* Wortwolken für LERA entwickelt. Abbildung 6.9 zeigt die stabilen Wortwolken für einen Abschnitt der *Histoire des deux Index*, der mittels CATview festgelegt wurde.

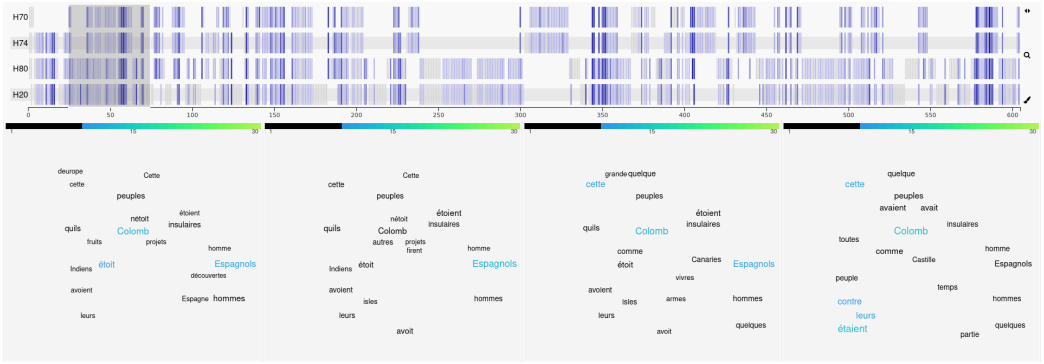


Abbildung 6.9: Mittels CATview wurde ein Abschnitt von 50 alignierten Textsegmenten gewählt und die relevantesten Wörter daraus in stabilen Wortwolken der vier Textfassungen der *Histoire des deux Indes* visualisiert.

In den stabilen Wortwolken taucht ein bestimmtes Wort immer an der gleichen Stelle auf, und zwar über alle Wortwolken der einzelnen Fassungen hinweg. Sie lösen damit ein entscheidendes Problem dieser Analyseform, denn das visuelle Verfolgen der Änderungen eines Begriffs ist schwierig bis unmöglich, wenn sich dessen Position in der Wortwolke bei jedem Neuzeichnen ändert. Durch die gemeinsam gewählten Positionen, wird zudem der fassungsübergreifende Textvergleich deutlich erleichtert. Abbildung 6.10 zeigt dies am Beispiel.

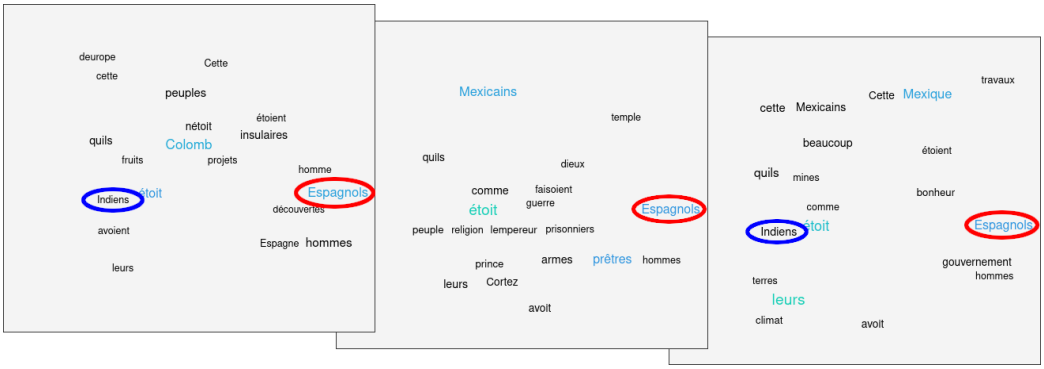


Abbildung 6.10: Drei stabile Wortwolken, deren Datengrundlage sich durch das Verschieben der Auswahl in CATview verändert hat. Die Wörter behalten dabei ihre Position, zum Beispiel „Espagnols“. Durch die Schriftgröße und -farbe wird die variierende Relevanz des Wortes im gewählten Textabschnitt visualisiert. Das Wort „Indiens“ verschwindet zwischenzeitlich aus der Menge der 20 relevantesten Wörter, taucht aber später an der selben Position in der Wortwolke wieder auf.

Um stabile Wortwolken für LERA zu realisieren, genügt es nicht, jedem Begriff eine fixe Position im Darstellungsbereich zuzuordnen, da dieser nicht beliebig groß ist und Wörter möglichst nicht übereinander gezeichnet werden dürfen. So besteht ein Zielkonflikt zwischen einer fixen Positionierung und der Platzeffizienz. Die Lösung dieses Problems wird im Folgenden kurz angerissen. Die Grundidee besteht darin, dass viele Wörter, die bezogen auf die jeweils aktuellen Einstellungen als relevant gelten, in der aktuell gewählten Größe des Auswahlfensters nie gemeinsam in einer Wortwolke auftreten können. Diese Wörter bilden Gruppen, die sich eine fixe Position teilen können. Das Konzept und der grundlegende Algorithmus zur Berechnung der Gruppen wurde in [63] vorgestellt. Wichtige Verbesserungen zur effizienten Berechnung, zur fassungsübergreifenden Positionierung sowie zum überschneidungsfreien und dennoch möglichst kompakten Layout werden in [32] diskutiert.

6.4 Explorative Analyse im Gesamtkorpus

Die synoptische Gegenüberstellung in LERA samt der bereits diskutierten Visualisierungen und ihrer Verknüpfung zur explorativen Analyse wurde für Einsatzszenarien mit drei bis acht Textfassungen entwickelt und optimiert. In Editionsvorhaben, wie beispielsweise der Edition von *Keter Shem Ṭov*, stehen die Editor:innen allerdings mitunter vor der Aufgabe, gleich Dutzende Textfassungen zu analysieren. Für solche Anwendungsfälle wurde mit der Korpussynopse eine weitere Komponente für LERA entwickelt, die gleichzeitig alle Textfassungen – sprich das gesamte Korpus – vergleichend visualisieren kann. Die Grundidee dahinter ist das Ermöglichen eines Überblicks über das Gesamtkorpus, um Gruppen ähnlicher Textfassungen zu identifizieren und interessante Varianten bereits vor dem detaillierten, synoptischen Vergleich erkennbar zu machen. Die grundlegende Idee sowie Umsetzung werden auch ausführlich in [115] diskutiert, aus dem einige Formulierungen für diesen Abschnitt übernommen wurden.

Die Korpussynopse fügt der Kollationierung in LERA damit auch eine optionale, dritte Vergleichsstufe auf Fassungsebene hinzu, mit der eine interessante Menge von Textfassungen aus dem Gesamtkorpus ausgewählt werden kann, bevor die Anwendung der Vergleiche auf Segment- und Tokenebene folgen.

Die Nutzeroberfläche der Korpussynopse besitzt dabei drei Teile. Der oberen Teil besteht aus einer graphischen Auflistung aller Textfassungen des Korpus, die hier aus- oder abgewählt werden können, um so die Grundlage für die folgenden Visua-

lisierungen festzulegen. Standardmäßig sind dabei alle Textfassungen ausgewählt.
Die eigentliche Korpussynopse befindet sich im mittleren Teil, in Form dreier interaktiver Visualisierungen, die im weiteren Verlauf des Unterkapitels vorgestellt werden. Dabei handelt es sich zum einen um eine modifizierte CATview-Übersichtsleiste für das Gesamtkorpus und zum anderen um einen interaktiven Federspanngraphen sowie ein Sehnendiagramm, die dynamisch für die getroffene Auswahl von Textfassungen erzeugt werden.
Der untere Teil ähnelt dem oberen. Hier findet sich ein zweiter Auswahlbereich sowie eine Schaltfläche, mit der ein LERA typischer, synoptischer Textvergleich der ausgewählten Textfassungen generiert werden kann.

6.4.1 Modifizierte Übersichtsleiste

In der Korpussynopse steht eine angepasste und erweiterte CATview als wählbare Visualisierungsform bereit. Darin wird eine Alignierung der Textsegmente zwischen allen beziehungsweise einer getroffenen Auswahl der Textfassungen dargestellt. Sie dient damit als Einstiegspunkt, um die Struktur der Textfassungen des gesamten Korpus überblicken zu können. Abbildung 6.11 zeigt die Visualisierung exemplarisch für ein Korpus von 70 Textfassungen von *Keter Shem Ṭov*.

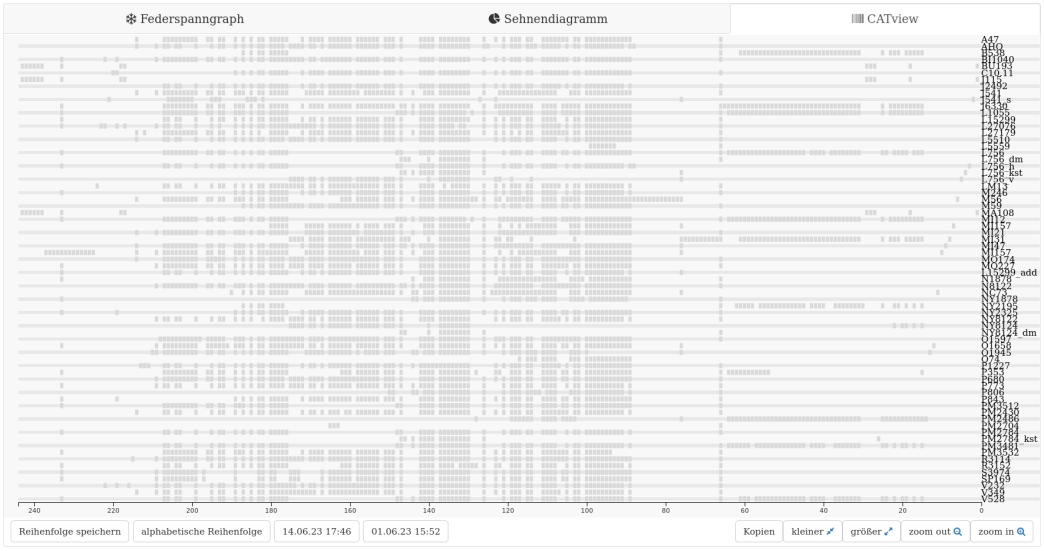


Abbildung 6.11: CATview-Übersichtsleiste mit 70 Textfassungen von *Keter Shem Ṭov* als Bestandteil von LERAs Korpussynopse.

CATview wurde für die Einbettung in die Korpussynopse in einigen Punkten modifiziert. So erlaubt die Übersichtsleiste eine Umsortierung der Textfassungen mittels der in Abschnitt 6.2 bereits beschriebenen Drag'n'Drop-Funktion. Die resultierende Reihenfolge kann via ergänzender Schaltfläche gespeichert und verschiedene Zwischenstände geladen werden. Weitere Schaltflächen erlauben, das Anzeigen von kopierten Segmenten ein- und auszuschalten, die für die Visualisierung verwendete Höhe zu verändern sowie innerhalb von CATview zu zoomen (als Alternative zum Mausrad).

6.4.2 Federspanngraph

Die Korpussynopse enthält als weitere Visualisierungsform einen interaktiven Federspanngraphen, dessen Knoten die Textfassungen des Korpus repräsentieren. Kanten zwischen den Knoten werden auf Basis eines Ähnlichkeitsmaßes gezogen, das auf die dazugehörigen Textfassungen angewendet wird, wobei die Nutzer:innen interaktiv über einen doppelten Schieberegler den Beginn und das Ende des gültigen Wertebereichs festlegen können. Die Kanten besitzen entsprechend des Ähnlichkeitsmaßes ein Gewicht, das die gezeichnete Dicke der Kanten festlegt: Je ähnlicher sich zwei Textfassungen sind, desto dicker wird die Kante zwischen den repräsentierenden Knoten.
Darauf aufbauend werden zwei Kräfte simuliert, die das Layout des Federspanngraphen dynamisch bestimmen. Die erste ist eine Anziehungskraft zwischen den, durch eine Kante verbundenen Knoten, die mit größerem Kantengewicht zunimmt. Hinzu kommt als Zweites eine Abstoßungskraft, die wiederum mit abnehmendem Ähnlichkeitswert zunimmt und dadurch insbesondere unverbundene Knoten voneinander separiert. Die beiden Kräfte schaffen so eine dynamische Struktur, in der ähnliche Textfassungen dazu neigen, nahe beieinander zu liegen, während zwischen unähnlichen Textfassungen größere Abstände entstehen. Das führt wiederum zur Bildung von Clustern ähnlicher Textfassungen innerhalb dieser Struktur. So können mithilfe des Federspanngraphen erste Einblicke in das Korpus gewonnen werden, um beispielsweise Gruppen von Textfassungen oder Außenseiter zu identifizieren. Abbildung 6.12 zeigt einen in LERAs Korpussynopse generierten Federspanngraphen für 70 Textfassungen von *Keter Shem Ṭov*.
Der Federspanngraph in LERA wird dynamisch für eine durch die Nutzer:innen gewählte Datengrundlage erstellt. Die Implementierung basiert, wie bei CATview, auf d3.js und ist interaktiv gehalten. Knoten können mit der Maus bewegt werden, so-

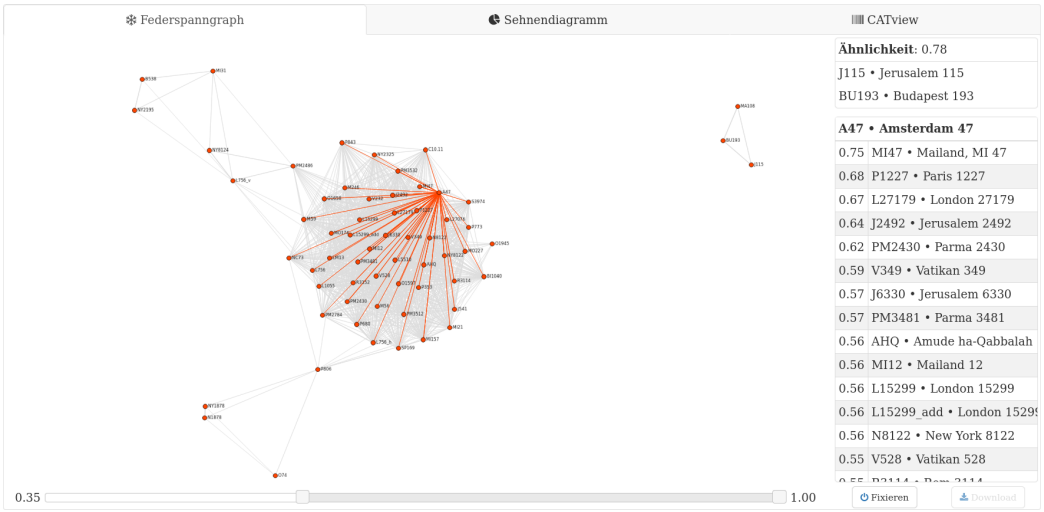


Abbildung 6.12: Federspanngraph für 70 Textfassungen von *Keter Shem Ṭov*, die durch die Knoten repräsentiert werden. Zwischen zwei Knoten besteht eine Kante, wenn der Ähnlichkeitswert (Jaccard-Koeffizient) zwischen den dazugehörigen Textfassungen im Bereich von 0,35 und 1,0 liegt. Dieser Wertebereich wurde mittels des Schiebereglers unten im Bild festgelegt. Die simulierten Anziehungs- und Abstoßungskräfte sorgen schließlich für das Layout des Graphen. Zudem wurde mittels eines Klicks auf den dazugehörigen Knoten eine Textfassung (Sigle *A47*) im Graphen hervorgehoben sowie Zusatzinformationen, wie die Ähnlichkeit zu allen anderen Textfassungen, eingeblendet.

dass ein Eindruck von den wirkenden Kräften beziehungsweise den Ähnlichkeiten vermittelt wird. Zudem erleichtern eine Zoom-Funktion sowie die Möglichkeit, den Bildschirmausschnitt zu verschieben, die Orientierung. Mittels des erwähnten Schiebereglers kann ein Bereich bezüglich des Ähnlichkeitsmaßes festgelegt werden, für den Kanten zwischen den Textfassungen gezogen werden. So lassen sich wahlweise Strukturen zwischen sehr ähnlichen, aber auch sehr unterschiedlichen Textfassungen untersuchen. Mittels eines Maus-Hover-Effekts an Knoten und Kanten werden optional weitere Informationen zur entsprechenden Textfassung (voller Titel) beziehungsweise dem Paar von Textfassungen (Ähnlichkeitswert) angezeigt. Zudem kann ein Knoten angeklickt werden, was ihn samt seiner Kanten im Graphen hervorhebt sowie eine Tabelle einblendet, welche alle anderen, absteigend nach ihrem Ähnlichkeitswert sortierten Textfassungen auflistet.

Die Wahl des Ähnlichkeitsmaßes ist eine entscheidende Variable für diese Untersuchungsmethode und stark abhängig von den verfolgten Forschungsfragen. Im Beispiel kam der Jaccard-Koeffizient, das heißt der Anteil der in beiden Textfassungen gemeinsam vorkommenden Wörter, zum Einsatz. Dieser ist eine geeignete Metrik, um eine vielfach geänderte Wortwahl zu erkennen, lässt aber strukturelle Änderungen, wie umfangreiche Erweiterungen mit einem ähnlichen Vokabular oder Transpositionen von Textsegmenten, unberücksichtigt. Andere Maße, wie beispielsweise der Anteil an übereinstimmenden Segmenten laut Alignierung zur Bewertung der strukturellen Ähnlichkeit, sind möglich, im aktuellen Prototyp jedoch noch nicht verfügbar.
Die algorithmische Komplexität eines Ähnlichkeitsmaßes ist ein zu berücksichtigender Faktor. Durch den notwendigen, paarweisen Vergleich der Textfassungen steigt mit einem wachsenden Korpus die Anzahl der Vergleiche drastisch an: Für n Textfassungen gibt es $½ \cdot n \cdot (n-1)$ zu vergleichende Paare von Textfassungen. Daraus ergibt sich ein Problem für die Reaktionsgeschwindigkeit des Systems. Allerdings, im Gegensatz zu den anderen in LERA durchgeführten Vergleichen – bei denen die gegebene Menge von Textfassungen gleichrangig verglichen wird, das heißt, die Textvarianten unmittelbar zwischen allen Fassungen erkannt werden müssen – können zwei Textfassungen für den Federspanngraphen unabhängig vom Rest des Korpus betrachtet werden, um ihren Ähnlichkeitswert zu ermitteln. So besteht eine angedachte, aber im aktuellen Prototypen noch nicht umgesetzte Lösung für dieses Problem in der Vorberechnung: Immer wenn eine Textfassung geändert oder dem Korpus hinzugefügt wird, berechnet das System im Hintergrund die Ähnlichkeiten zu allen anderen Textfassungen und speichert die Ergebnisse in LERAs Datenbank. So stünden diese bereits zur Verfügung, wenn die Korpussynopse von den Nutzer:innen aufgerufen wird.

6.4.3 Sehnendiagramm

LERAs Korpussynopse bietet als dritte Visualisierungsform ein interaktives Sehnendiagramm an, welches ebenfalls mittels d3.js realisiert wurde. Das Sehnendiagramm arbeitet dabei auf der selben Datengrundlage wie der Federspanngraph und bietet eine ähnliche Nutzeroberfläche, was unter anderem das Verhalten und die Gestaltung beim Einblenden von Zusatzinformationen zu den Textfassungen und ihren Ähnlichkeitswerten zueinander beinhaltet. Abbildung 6.13 zeigt dies am Beispiel.

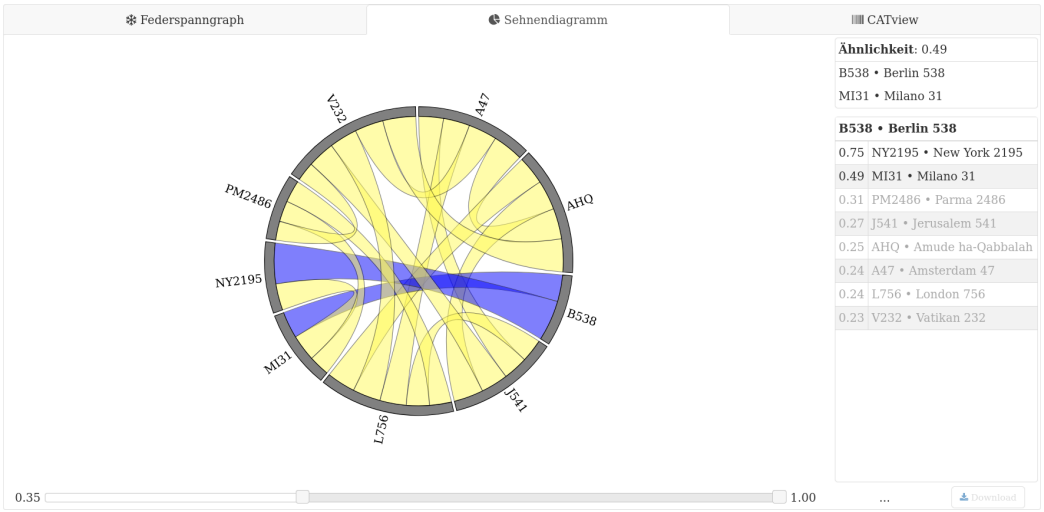


Abbildung 6.13: Sehnendiagramm für neun Textfassungen von *Keter Shem Ṭov*. Der Klick auf die Textfassung B538 zeigt deren Beziehungen zu den anderen Textfassungen in Abhängigkeit der mittels des Schiebereglers zugelassenen Ähnlichkeitswerte (Jaccard-Koeffizient zwischen 0,35 und 1,0).

Diese Visualisierungsform eignet sich durch die Darstellung der Kantengewichte in visuell gut zu unterscheidenden Breiten für die Untersuchung der Ähnlichkeiten ausgehend von einer Textfassung. Allerdings sollte die Anzahl der Textfassungen dabei nicht zu groß gewählt werden, da der verfügbare Platz durch ihre Anordnung als Kreissegmente beschränkt ist. So bietet sich das Sehnendiagramm vorrangig als ergänzende Untersuchungsmethode an, nachdem die Auswahl der Textfassungen durch die Analyse mit den beiden anderen Visualisierungsformen bereits eingeschränkt wurde.

Kapitel 7

Fazit

Im nun folgenden, abschließenden Kapitel dieser Dissertation werden zunächst die erzielten Ergebnisse zusammengefasst. Das beinhaltet sowohl einen ausführlichen Abgleich der Zielsetzungen mit den tatsächlich realisierten Software-Lösungen und damit die Beantwortung der gestellten Forschungsfragen als auch eine Einordnung der aktuellen Nutzerbasis von LERA. Abschließend folgt ein Ausblick über weitere geplante Verbesserungen sowie mögliche, teils mit neuen Forschungsfragen verknüpfte Erweiterungen der Software.

7.1 Zusammenfassung

In Abschnitt 2.3 wurden anhand verschiedenartiger Editionsprojekte sechs wesentliche Bedarfe und Anforderungen an digitale Werkzeuge für die Kollationierung unterschiedlicher Textfassungen eines Werkes abgeleitet. Ein Abgleich mit bestehenden Methoden und verfügbarer Software in der Domäne des (philologischen) Textvergleichs in Abschnitt 3.2 zeigte auf, dass diese Bedarfe nur teilweise abgedeckt werden. Mit LERA wurde ein Tool geschaffen, das gleich mehrere dieser aufgezeigten Desiderate annimmt, Lösungen umsetzt und so Nutzer:innen die editorische Arbeit erleichtert. Im Folgenden werden die aus den ermittelten Anforderungen extrahierten Zielsetzungen für LERA nochmals betrachtet und die in dieser Dissertation beschriebenen Lösungen zusammengefasst. Begleitend zu dieser Zusammenfassung folgt eine kurze Einordnung der Nutzerbasis von LERA, welche den Erfolg der Software und damit die Akzeptanz der implementierten Lösungen in der Fachcommunity unterstreicht.

7.1.1 Abgleich mit den Zielsetzungen

Vergleich von mehreren Textfassungen (I)

Die erste Zielsetzung an LERA war die Möglichkeit zum Vergleich von mehreren Textfassungen. Die Handhabung von mehr als zwei Textfassungen wurde bei der Entwicklung von Anfang an berücksichtigt und alle Bestandteile der Software, wie die implementierten Algorithmen oder Visualisierungen, entsprechend entworfen und umgesetzt.
Dabei wurde aufgrund des ermittelten Bedarfs die Entscheidung getroffen, statt eines Leitquellenprinzips einen gleichrangigen Vergleich der Textfassungen zu realisieren[1], was die verschiedenen Vergleichsebenen (Token, Segmente, Fassungen) betrifft und einen deutlichen Unterschied zu vielen anderen Werkzeugen in dieser Domäne darstellt. Ergänzend zum gleichrangigen Vergleich gibt es in LERA allerdings auch einen wählbaren Farbmodi, welcher die Editieroperationen und damit direkt die von den Nutzer:innen gewählte Chronologie der Textfassungen visualisiert.
Um den Umgang mit vielen (Hunderten) Textfassungen zu ermöglichen, wurden in LERA zusätzliche Komponenten und Visualisierungen geschaffen, welche den klassischen Anwendungsfall von bis zu acht Textfassungen erweitern[2]. Dazu gehören die alternative Darstellung des Kollationierungsergebnisses als Partitursynopse sowie die explorative Analyse via Korpussynopse.

Vergleich von umfangreichen Textfassungen (II)

Die zweite Zielsetzung an LERA war die Möglichkeit zum Vergleich von sehr langen Textfassungen, beispielsweise kompletter Bücher wie im Falle der *Histoire des deux Indes*. Dies konnte durch die Auftrennung der Kollationierung in eine Alignierung größerer Textsegmente mit einem daran anschließenden Detailvergleich auf Tokenebene realisiert werden. Dieses hierarchische Vorgehen mittels zweier Vergleichsstufen spiegelt sich dabei nicht nur in getrennt voneinander aufgerufenen, für

[1] Die gegebene Reihenfolge der Textfassungen kann das Vergleichsergebnis der in LERA integrierten Algorithmen allerdings beeinflussen, nämlich genau dann, wenn zwei oder mehr gleichwertige Lösungen für einen algorithmischen Teilschritt existieren und automatisch eine davon gewählt wird.

[2] Die Anzahl wurde nie strikt limitiert, doch wird die klassische spaltensynoptische Darstellung durch die eingeschränkte Bildschirmbreite irgendwann unhandlich.

ihre Teilaufgabe optimierten Algorithmen wider, sondern auch in der Gestaltung der Nutzeransicht. So bietet die synoptische Gegenüberstellung von Textsegmenten – üblicherweise Sätze oder Absätze – den Nutzer:innen genügend Kontext zum Einordnen der darin enthaltenen Textvarianten bei gleichzeitiger Beschränkung der Länge, um beim Lesen leicht überschaubar zu bleiben.
Die Handhabung umfangreicher Textmengen wird zudem durch die integrierten Visualisierungen unterstützt, indem diese die Daten aggregiert darstellen. Zusätzlich erleichtert CATview dabei die Navigation in der synoptischen Volltextrepräsentation mittels einer wechselseitigen Verknüpfung beider Komponenten.

Umgang mit komplexen Textvarianten (III)

LERA sollte als dritte Zielsetzung den Umgang mit verschiedenartigen und teils komplexen Textvarianten erlauben. Dazu wurde zum einen mittels eines Filtersystems die Möglichkeit geschaffen, jederzeit einzelne Klassen von Textvarianten beim detaillierten Vergleich auf Tokenebene durch geeignete Normalisierung oder Mustererersetzung zu ignorieren. Einige Filter können zudem separat auf die Darstellung während der synoptischen Volltextrepräsentation angewandt werden, um so die Lesbarkeit zu erhöhen.
Zum anderen wurde ein besonderer Fokus auf die Handhabung struktureller Unterschiede zwischen den Textfassungen gelegt. Durch Aufteilung der Kollationierung in Segment- und Tokenebene werden großflächige Überarbeitungen ganzer Textsegmente algorithmisch sowie in der Darstellung handhabbar. Das gilt auch für Umstellungen in Form von transponierten Textsegmenten, die den Nutzer:innen durch die gesonderte Darstellung als kopierte Segmente (siehe Abschnitt 4.2.5) im Volltext sowie in der Übersichtsleiste veranschaulicht werden. LERA bietet darüber hinaus die Möglichkeit, in die (automatisch bestimmte) Segment-Alignierung einzugreifen, was ebenfalls als Folge struktureller Unterschiede zwischen den Textfassungen notwendig werden kann.

Nutzerfreundlichkeit des Systems (IV)

Die vierte Zielsetzung an LERA war es, eine möglichst hohe Nutzerfreundlichkeit des Systems zu erreichen und so die Anwendungsschwelle möglichst niedrig zu halten. Ein wesentlicher Baustein dafür ist die Realisierung von LERA als Webanwendung, da so eine Möglichkeit zur Nutzung ohne eigene Installation der Software und

somit ohne die Notwendigkeit eigener performanter Hardware geschaffen wurde, was die Anwendungsschwelle bereits deutlich senkt. So wird LERA als Software-as-a-Service für interessierte Forscher:innen aus Editionsprojekten bereitgestellt, sodass die Nutzeroberfläche unkompliziert über gängige Browser aufgerufen werden kann, während das technisch anspruchsvolle Hosting eines Webservers vor den Nutzer:innen verborgen wird.

Zudem wurde für LERA eine grafische Nutzeroberfläche realisiert, welche anhand der von Susanne Schütz erstellen Editionsprobe zur *Histoire des deux Indes* eine konkrete Arbeitsweise aus den Geisteswissenschaften nachahmt und so die Funktionalitäten der Software intuitiv für den anvisierten Nutzerkreis bereit stellt. In die Weiterentwicklung LERAs ist zudem das kontinuierliche Feedback durch Anwender:innen aus den Geisteswissenschaften eingeflossen. Während eine umfangreiche Studie zur angestrebten Usability noch aussteht, bestätigt zumindest anekdotische Evidenz[3] die Nutzerfreundlichkeit des Systems.

Des Weiteren trägt die Bereitstellung verbreiteter Ein- und Ausgabeformate zur breiteren Nutzbarkeit von LERA bei. So wird mit TEI-XML zum Import von Textfassungen der de-facto-Standard in den digitalen Editionswissenschaften unterstützt. Um dabei auch den verschiedenen Auszeichnungen der individuellen Editionsprojekte gerecht zu werden, wurde die Möglichkeit der automatischen Vorverarbeitung der Textfassungen beim Import via XSLT integriert. LERA wurde zudem mit umfangreichen Möglichkeiten zum Export der erzielten Ergebnisse ausgestattet. Dazu gehören Exportformate, die direkt für die Präsentation verwendet (HTML, PDF), als Grundlage einer Printedition genutzt (TEX) sowie in maschninenlesbarer Form von anderen digitalen Werkzeugen weiterverarbeitet (JSON, XML) werden können.

[3] Zum Beispiel: „LERA is an impressively coherent suite of tools for text alignment and collation which allows the user to flexibly combine tools and parameters for individual use cases“ in [129], „Through the segmentation system that we have designed, and thanks to the LERA platform that we are using, the process of comparing manuscript versions of select passages has become easier, clearer, and more precise.“ in [54] sowie „I've used LERA now to support my edition of *The Adventures of Sherlock Holmes* and found it to be excellent, and very well suited to our purpose. It is sufficiently intuitive that I found I could work out how to use it after an hour's experimentation. In terms of functionality, I particularly like the different ways you can visualise variants, e.g. using colour codes or as endnotes. Uploading documents is extremely easy, and the algorithmic text matching works brilliantly.“ aus privater Korrespondenz mit Andrew Glazzard vom *The Edinburgh Conan Doyle Project* am 21.03.2021.

Interaktivität des Systems (V)

Die fünfte Zielsetzung für LERA war die Interaktivität des Systems, für dessen Erreichen die wesentlichen Funktionalitäten interaktiv gestaltet wurden. Dazu gehört die Flexibilität der Parameterwahl beim Bestimmen von Textvarianten, welche bei klassischen Printeditionen sowie mitunter bei digitalen Umsetzungen fehlt, bei der Entwicklung von LERA hingegen forciert und schließlich realisiert wurde. So können wesentliche Vergleichsparameter, wie die Auswahl der Textfassungen oder die gewünschte Normalisierung, dynamisch über die Nutzeroberfläche angepasst werden. Hinzu kommt die genannte Eingriffsmöglichkeit bei der Segmentalignierung, welche ebenfalls jederzeit über die grafische Nutzeroberfläche angepasst werden kann.
In LERA wurden verschiedene, teils neu entwickelte Visualisierungen integriert, welche die Interaktivität des Gesamtsystems erhöhen. Neben der Navigation in großen Textmengen mittels der Übersichtsleiste CATview standen dabei insbesondere innovative Möglichkeiten zur explorativen Analyse von Textvarianten im Fokus. So schafft die in Abschnitt 6.3.2 beschriebene Kombination aus CATview, stabilen Wortwolken und der Suchfunktion einen gänzlich neuen und dabei interaktiven Zugang zu den Textvarianten. Auch auf der Vergleichsebene ganzer Textfassungen wurde mit der Korpussynopse ein interaktiver Zugang geschaffen.
Als notwendige Grundlage für die Interaktivität des Systems war es notwendig, LERA – beziehungsweise spezifische Funktionalitäten – als Echtzeitanwendung zu realisieren. Kurze Wartezeiten wurden dank effizienter Algorithmen zur Kollationierung und der Optimierung der Software an vielen Stellen im Front- und Backend erreicht, wie exemplarisch anhand einiger Techniken in Abschnitt 4.4 beschrieben wurde.

Generizität des Systems (VI)

Die sechste und finale Zielsetzung war es schließlich, mit LERA ein generisches System zu schaffen, das in einem möglichst breiten Spektrum von Editionsprojekten, welche die Kollationierung verschiedener Textfassungen verfolgen, angewendet werden kann.
Als Grundlage dafür bietet LERA Lösungen für die vier, in Abschnitt 2.2.3 diskutierten Zielsetzungen der Kollationierung. Zum Ersten erleichtert LERA das Transkribieren von Textfassungen, insofern die Textfassungen eine gewisse Ähnlichkeit

aufweisen, wie sich bei verschiedenen Editionsprojekten gezeigt hat. Bei geeigneter Wahl des Farbmodus (*Editieroperationen* oder *Einzelvorkommen*) fallen durch das farbliche Hervorheben der Textvarianten in der synoptischen Volltextrepräsentation enthaltene Transkriptionsfehler leicht ins Auge. So stehen Transkription und Vergleich der Textfassungen in einer Wechselwirkung, die das Endresultat verbessern kann. Zum Zweiten ermöglicht LERA das Dokumentieren der Unterschiede zwischen verschiedenen Textfassungen. Die Textvarianten werden bestimmt und können über die Nutzeroberfläche angezeigt oder in verschiedenen Formaten exportiert werden. Das integrierte Filtersystem erlaubt es dabei, wiederkehrende Unterschiede, beispielsweise in Folge einer Rechtschreibreform oder durch Varianten von Eigennamen, auszunehmen, wenn diese stattdessen mittels einer zusammenfassenden Anmerkung in der entstehenden Edition dokumentiert werden sollen. Ein Vorteil der Nutzung von LERA ist dabei auch, dass sich diese Editionsrichtlinien nachträglich leicht ändern lassen, indem bestehende Filter gewählt oder neue ergänzt werden, ohne dass jahrelange Zusatzarbeit entsteht. Zum Dritten unterstützt LERA Forschende beim Aufzeigen des Abstammungsverhältnisses von Textfassungen und dem darauf aufbauenden Nachzeichnen der Entstehungs- und Änderungshistorie innerhalb von Traditionslinien. So können strukturelle Unterschiede zwischen Textfassungen gut durch Segmentaligierung und Übersichtsleiste erkannt werden. Diese Ähnlichkeitsnanalyse auf Makroebene ermöglicht eine erste Einordnung und ist ebenso in der Korpussynopse möglich. Wesentlich sind allerdings oft die Evidenzen auf Mikroebene, zum Beispiel markante Stellen, die nur in wenigen Textfassungen oder in gewissen Konstellationen vorkommen. Sie können unterstützend mittels Filtern und den extra aus diesem Hintergrund heraus integrierten Farbmodi (*Einzelvorkommen*, *Paare* und *Gruppen*) hervorgehoben werden. Da mitunter auch Einzelvarianten oder -merkmale, zum Beispiel die gewählten Akzente, entscheidend sein können, wurden Filter optional gehalten. Zum Vierten unterstützt das integrierte Filtersystem Editor:innen beim Erstellen einer historisch-kritischen Edition. Da das Hauptinteresse hier auf den inhaltlichen Änderungen beziehungsweise ihrer kritischen Einordnung in den historischen Kontext liegt, können die für die verfolgten Forschungsfragen irrelevanten Textvarianten ausgeblendet werden. Der Textvergleich wird durch diese Normalisierung wesentlich übersichtlicher, was die editorische Arbeit erleichtert.
Für die Generizität des Systems ist auch eine breite Unterstützung der Sprachvielfalt wichtig. LERA arbeitet intern mit Unicode und erlaubt so die Verarbeitung der meisten Alphabete. Zusätzlich ist die Einbindung eigener Schriftarten möglich, was

ursprünglich für das Editionsprojekt zum *Manual de confesores y penitentes* zur Darstellung von mittelalterlichem Portugiesisch integriert und beispielsweise für das traditionelle Schriftsystem Vietnams – Chữ Nôm – im Rahmen der Untersuchung eines vietnamesischen Gedichts wiederverwendet wurde [89]. Eine Besonderheit von LERA stellt die Unterstützung der Schreibrichtung Rechts-nach-Links dar, welche in vergleichbaren Werkzeugen meist fehlt (siehe Abschnitt 3.2.6). Dabei wird die Schreibrichtung sowohl in der Darstellung des Fließtextes berücksichtigt, als auch bei der Anordnung verschiedener Elemente der Nutzeroberfläche miteinbezogen, um den unterschiedlichen Lesegewohnheiten gerecht zu werden. Die Sprachauswahl von LERA umfasst 14 Sprachen, wobei diese Zahl durch die modularisierte Software-Architektur leicht erweiterbar ist und mit neu hinzukommenden Editionsvorhaben steigen wird.
Zuletzt weist LERA eine mit den Anforderungen gewachsene, umfangreiche Konfigurierbarkeit für verschiedene Einsatzszenarien auf, wie in Abschnitt 4.4.1 bereits diskutiert wurde.

7.1.2 Nutzerbasis von LERA

Bereits mit dem Ende der ersten Entwicklungsphase von LERA anhand der *Histoire des deux Indes* im Rahmen des Projekts *SaDA - Semi-automatische Differenzanalyse von komplexen Textvarianten* (2012-2015) wurde die Software von ersten externen Editionsvorhaben angefragt und wird teilweise noch heute genutzt. Dazu gehören beispielsweise das Langzeitprojekt *AnonymClassic* zur Erforschung von *Kalīla and Dimna*[4] (siehe Abschnitt 2.1.3) und das Projekt *HyperAzpilcueta*, bei dem ein internationales Team an einer digitalen Edition des in drei verschiedenen Sprachen verfassten *Manual de confesores y penitentes* arbeitet (siehe Abschnitt 2.1.4). Zudem seien mit *Hannah Arendt. Kritische Gesamtausgabe*[5] und der *Edinburgh Edition of the Works of Arthur Conan Doyle* [6] zwei weitere langfristige Editionsvorhaben der Werke bekannter Autor:innen genannt, welche für die Kollationierung der Textfassungen LERA einsetzen.

[4] Erste, mit der Unterstützung von LERA erstellte Ergebnisse wurden bereits unter `https://kd-preview-edition.geschkult.fu-berlin.de/collations` (besucht am 25.07.2024) online gestellt.

[5] Teil der digitalen Edition unter `https://hannah-arendt-edition.net` (besucht am 19.12.2023) ist auch eine eigene LERA-Instanz, mit der aktuell drei Werke synoptisch von den Nutzer:innen verglichen werden können.

Allein in den letzten drei Jahren (2021 bis 2023) gab es in Summe 56 Anfragen externer Forscher:innen zur Nutzung von LERA, für welche je eine eigene Instanz der Software bereit gestellt wurde. Die Hintergründe sind dabei verschieden und reichen von der Validierung von Hypothesen im Rahmen von Fachartikeln[6] oder Abschlussarbeiten[7] über Editionsvorhaben einzelner Wissenschaftler:innen[8] bis hin zu den bereits genannten Langzeitprojekten mit größeren Teams.
Hinzu kommen gut ein Dutzend von der Arbeitsgruppe intern genutzte Instanzen für studentische Abschlussarbeiten im Themenbereich des Textvergleichs oder im Rahmen von direkt begleiteten Editions-[9] und Forschungsprojekten[10].

7.2 Ausblick

Die Entwicklung von LERA ist ein fortlaufender Prozess. Insbesondere durch die wachsende Anzahl von Editionsprojekten, in denen LERA zum Einsatz kommt und welche mitunter für die Software neuartiges Textmaterial untersuchen, entstehen neue Wünsche und erweiterte Anforderungen. Dieses Unterkapitel gibt einen Überblick über die noch geplanten sowie zusätzlich denkbaren Erweiterungen von LERA. Dabei gliedert sich die Auflistung in drei Kategorien. Zunächst werden die geplanten Erweiterungen genannt, welche die aktuell bestehenden Funktionalitäten abrunden. Die zweite Kategorie beschreibt naheliegende Erweiterungen in Form neuer Funktionalitäten, die mit überschaubarem Aufwand ergänzt werden können.

[6] Ein Beispiel hierfür ist ein 2022 erschienener Artikel von Alejandro García-Reidy, siehe [50], in dem er mithilfe von LERA zeigt, dass es sich bei der Lope de Vega zugeschriebenen Komödie *El cortesano embustero* tatsächlich um eine Textfassung von *La española* des Dramatikers Cepeda handelt.

[7] Ein Beispiel hierfür ist die 2021/2022 an der Universität Trier von Tamara Schuster bearbeitete und erfolgreich abgeschlossene Masterarbeit *Das digitale Rattennest – Digitalisierung, Transkription, Auszeichnung und Erschließung verschiedener Fassungen von Oskar Wöhrles „Das Rattennest" aus seinem literarischen Nachlass sowie Konzeptionierung und Implementierung einer digitalen Textedition*.

[8] Ein Beispiel ist die von Dr. Andrejka Žejn erstellte digitale Plattform „Izhodišča slovenske pripovedne proze" [157]. Für die Kollationierung und Visualisierung von sechs Textfassungen der Kurzgeschichte *Sreča v nesreči* wurde LERA genutzt, siehe `https://ispp.zrc-sazu.si/izdaje-srece-v-nesreci` (besucht am 28.09.2023).

[9] Wie der hybriden Edition der Werke Karl Gutzkows, siehe `https://gutzkow.uzi.uni-halle.de` (besucht am 28.09.2023).

[10] Beispielsweise zwei aufeinander aufbauende Forschungsprojekte im Kontext der Untersuchung des kabbalistischen Traktats *Keter Shem Ṭov* [99].

Schließlich werden als Drittes konzeptionelle Erweiterungen genannt, welchen einen gewissen Forschungsaufwand beinhalten und daher einen umfangreichen beziehungsweise schwer kalkulierbaren Aufwand implizieren.

7.2.1 Abrundung bestehender Funktionalitäten

Ein angedachter nächster Schritt ist die systematische Evaluation der grafischen Nutzeroberfläche über die Durchführung einer Nutzerstudie, um mögliche Schwachstellen der aktuellen Implementierung zu erkennen und so die Nutzerfreundlichkeit weiter zu erhöhen. Diese Studie ist zudem dazu gedacht, die im Rahmen dieser Dissertation mittels anekdotischer Evidenz gefolgerte hohe Nutzerfreundlichkeit von LERA zu prüfen. Durch die in den letzten Jahren stark angewachsene Nutzerbasis ist nun auch eine ausreichende Stichprobengröße möglich, um neben qualitativen (punktuellen) Ergebnissen auch zu quantifizieren.

Mit der in Abschnitt 6.4 vorgestellten Korpussynopse wurde eine neue Komponente von LERA zur Analyse sehr vieler Textfassungen anhand des Beispiels von *Keter Shem Ṭov* geschaffen. Die derzeitige Implementierung benötigt eine umfangreiche Vorberechnung, die jeweils für den aktuellen Datenbestand manuell angestoßen werden muss, in der Folge aber kurze Reaktionszeiten für die eigentliche visuelle Analyse der Daten ermöglicht. Diese Vorberechnung (für 70 Textfassungen benötigt sie um die 15 Minuten[11]) steht allerdings im Widerspruch mit dem Ziel kurzer Reaktionszeiten (siehe **V.**), lässt sich durch eine angedachte Erweiterung jedoch umgehen. So könnten die benötigten Daten auch automatisch vom System im Hintergrund berechnet und bereit gehalten werden. Eine Aktualisierung wäre immer dann notwendig, wenn neue Textfassungen ins System hochgeladen oder bestehende modifiziert werden.

Eine weitere sinnvolle Ergänzung ist eine Eingriffsmöglichkeit in den detaillierten Vergleich auf Tokenebene analog zu der bereits bestehenden Möglichkeit, manuell in die Segmentierung und Alignierung der Textsegmente einzugreifen. So bestehen zwar indirekte Eingriffsmöglichkeiten über die Filter und Farbmodi beim automatischen Bestimmen der Textvarianten, eine direkte Modifikation – Hinzu-

[11] Getestet wurde auf einem Laptop mit einem 64-bit-Betriebssystem (Ubuntu 22.04), vier Kernen mit einer Taktung von je 2,40GHz (Intel® Core™ i7-5500U) und acht GiB DDR3-Arbeitsspeicher mit einer Geschwindigkeit von 1600 MT/s.

fügen, Entfernen oder Verändern der als Textvariante markierten Textstellen über die grafische Nutzeroberfläche – ist momentan nicht möglich. Diese angedachte Erweiterung würde schließlich den mit LERA verfolgten semi-automatischen Ansatz vervollständigen, welcher den Editor:innen bei jedem Bearbeitungsschritt die finale Entscheidung zugesteht.

Eine kleine, aber für einige Editionsprojekte sehr nützliche Ergänzung von LERA wäre die Möglichkeit, Illustration in Form von Grafiken in den Fließtext einzubetten. Werke weisen mitunter verschiedenartige Illustrationen auf, beispielsweise Diagramme oder Zeichnungen, die sich zwischen den Textfassungen in ihrer Gestaltung oder gar dem Vorhandensein unterscheiden. Sie können für die Analyse durch die Editor:innen wie auch für die Leserschaft der entstehenden digitalen Edition von hohen Interesse sein. So ermöglicht LERA ihre Repräsentation derzeit im Fließtext durch einen symbolischen Indikator, der zudem eine Bildunterschrift beinhalten kann und als Sonderelement in den hochgeladenen Textdokumenten ausgezeichnet wird. An dieser Stelle wäre eine Verknüpfung mit einer Bilddatei denkbar, in der ein Faksimile der Illustration hinterlegt ist und beispielsweise beim Berühren mit der Maus als Popup anzeigt wird.

7.2.2 Erweiterungen naheliegender Funktionalitäten

Stellenkommentare sind ein wesentlicher Bestandteil historisch-kritischer Editionen und die Möglichkeit zur Kommentierung entsprechend eine naheliegende Erweiterung von LERAs Funktionsumfang. Es gibt bereits eine rudimentäre Kommentarfunktion, die es erlaubt, für ganze Textfassungen oder eine Menge von alignierten Segmenten mittels einer Textbox einen Kommentar zu hinterlegen. Für echte Stellenkommentare wäre allerdings eine umfangreichere Erweiterung der Nutzeroberfläche notwendig, mit der konkrete Textstellen – auch segmentübergreifend – markiert und kommentiert werden können. Die entstandenen Stellenkommentare müssten schließlich auch der Leserschaft über eine geeignete Einbettung im Fließtext oder als Teil des optional zuschaltbaren Apparats zugänglich gemacht sowie in der Download-Funktion berücksichtigt werden.

Abseits des eigentlichen Textinhalts oder eingebetteter Illustrationen unterscheiden sich die Textfassungen mitunter auch in gestalterischen Aspekten voneinander, wie beispielsweise einem abweichenden Seitenaufbau oder der Verwendung verschiede-

ner typografischer Elemente. Diese Textvarianten können ebenfalls forschungsrelevant sein und sollten perspektivisch durch die Kollationierungsalgorithmen, wenn auch nur optional, erkennbar werden. Die Grundlage dafür ist in LERA geschaffen, da bereits verschiedene Elemente des Seitenaufbaus sowie typografische Elemente eingelesen, verarbeitet und dargestellt werden können, auch wenn sie momentan beim Textvergleich unberücksichtigt bleiben.

Eine weitere naheliegende und besonders im Kontext größerer Editionsprojekte nützliche Erweiterung von LERA ist die Integration einer umfangreichen Nutzerverwaltung, die das Vergeben und Verwalten von Rollen mit unterschiedlichen Rechten erlaubt. Auch hierfür wurde die Grundlage durch die in Abschnitt 4.4.1 beschriebenen Konfigurationsmöglichkeiten der Software bereits geschaffen. So können mittels Konfiguration einzelne Funktionalitäten an- und abgeschaltet sowie eine Zuordnung von Datensätzen zu spezifischen Nutzer:innen (beziehungsweise ihren Browsern) vorgenommen werden, was als Teil einer Nutzerverwaltung wiederverwendet werden kann.

7.2.3 Konzeptionelle Erweiterungen

Ein teils geäußerter Wunsch und als zusätzliche Option denkbare Funktionalität wäre die Kollationierung nach dem Leitquellenprinzip. Da LERA grundlegend nach dem gleichrangigen Vergleichsprinzip arbeitet und entsprechend konzipiert wurde, wäre hierfür die Anpassung mehrerer Softwarekomponenten sowie die Ergänzung zusätzlicher Vergleichsalgorithmen notwendig. Letztere wären, da jeweils nur der paarweise Vergleich der Textfassungen mit der gewählten Leitquelle notwendig wäre, zumindest komplexitätstheoretisch trivial. Allerdings ist von einem gewissen Aufwand zur Anpassung der Datenhaltung sowie der Nutzeroberfläche auszugehen. Zur Wahl der Leitquelle könnte die in CATview bereits implementierte Funktionalität zur interaktiven Vertauschung von Textfassungen einbezogen werden.

Neben der in Abschnitt 4.2.5 beschriebenen Handhabung von Transpositionen auf Segmentebene fehlt in LERA noch der direkte Umgang mit Vertauschungen auf Tokenebene, beispielsweise der Umstellung von Wörtern in einem Satz. Zwar liefert die Damerau-Levenshtein-Distanz als Erweiterung der in LERAs Algorithmus zur detaillierten Differenzanalyse verwendeten Levenshtein-Distanz bereits eine Grundlage dafür, eine neue Operation *:trans* zu integrieren, doch ist diese nur für zwei

Texte definiert und betrachtet zudem auch nur die Vertauschung zweier benachbarter Token. Die Frage, ob und wie effizient beliebige Transpositionen im Kontext der gewählten Modellierung des Textvergleichsproblems nach dem gleichrangigen Vergleichsprinzip integriert werden können, stellt eine offene Forschungsfrage dar, wobei verwandte Arbeiten einen Sprung der Komplexität vermuten lassen (siehe Abschnitt 3.2.3).

Der Textvergleich in LERA findet mit Ausnahme einiger optionaler Normalisierungen auf der Textoberfläche, das heißt der in den Token vorkommenden Zeichen und ihrer Anordnung, statt. Dem gegenüber steht ein semantischer Vergleich, also ein Abgleich der Bedeutungsähnlichkeit, welcher ebenso denkbar und für manche Anwendungsszenarien, wie das Ausblenden von synonymen Textvarianten zur Untersuchung von Bedeutungsverschiebungen bei stark überarbeiteten Textfassungen, sinnvoll wäre. Dies könnte in LERA durch die Einbindung zusätzlicher computerlinguistischer Ressourcen geschehen. Für die Erkennung von Synonymen wären das beispielsweise Worteinbettungen[12] oder Wortnetze[13]. Auch für den semantischen Vergleich von Phrasen existieren erprobte Methoden, beispielsweise der Einsatz von Distanzmaßen wie der Word Mover's Distance [83], welche ebenfalls auf Basis von Worteinbettungen arbeitet.
Allerdings gibt es dabei einige Hürden zu überwinden. So sind Ressourcen, wie die benötigten Worteinbettungen, nicht für alle Sprachen in ausreichender Qualität verfügbar. Worteinbettungen sind zudem sehr gewichtige Datenstrukturen, was durch die vielen in LERA zu unterstützenden Sprachen zusätzlich problematisch ist. Darüber hinaus ist neben der Effektivität der genannten Ansätze auch die Effizienz ein zu untersuchender Aspekt. So benötigt beispielsweise die klassische Word Mover's Distance eine kubische Laufzeit, welche im praktischen Einsatz aber gegebenenfalls umgangen werden kann [116].

Eng verwoben mit dem Problem eines semantischen Vergleichs ist das Problem eines sprachübergreifenden Vergleichs, wie er in vielen Editionsvorhaben – siehe zum Beispiel *HyperAzpilcueta* (Abschnitt 2.1.4) oder *Hannah Arendt. Kritische Gesamt-*

[12] So stellt beispielsweise die Open Source Bibliothek *fastText* trainierte Wortvektoren für 157 Sprachen bereit, siehe `https://fasttext.cc/` (besucht am 11.10.2023).

[13] Die *Global WordNet Association* listet zum Beispiel freie Wortnetze für die meisten der von LERA unterstützten Sprachen, siehe `http://globalwordnet.org/resources/wordnets-in-the-world/` (besucht am 11.10.2023).

ausgabe (Abschnitt 2.1.5) – hilfreich oder gar notwendig für den Erfolg des Projekts ist. LERA unterstützt diesen zumeist manuell durchgeführten Teil der Untersuchung bisher nur durch eine Programmierschnittstelle, mit der eigene Segmentierungs- und Alignierungsheuristiken ans System angebunden werden können und welche bereits für den im Projekt *HyperAzpilcueta* entwickelten Ansatz [148] genutzt wurde. Doch im Allgemeinen ist das Problem der automatischen, sprachübergreifenden Kollationierung im Kontext der Editionsphilologie noch ein offenes Forschungsdesiderat, welches im Rahmen des Projekts *Semi-automatische Kollationierung verschiedensprachiger Fassungen eines Textes* [2] in den kommenden Jahren untersucht wird und dessen Ergebnisse in LERA einfließen werden.

Eine weitere, von LERA nur in Ansätzen berührte Vergleichsebene stellt der visuelle Vergleich von Textfassungen dar. Das beinhaltet zum einen die bereits oben erwähnte Integration von Illustrationen, die neben der reinen Präsentation für die Nutzer:innen auch mittels Bilderkennung auf abweichende und übereinstimmende Eigenschaften überprüft werden könnten. Zum anderen ist auch ein Mechanismus zur Einbettung von Faksimiles der Textfassungen selbst wünschenswert, sodass während der Arbeiten in LERA anhand der digitalen Transkriptionen auch stets das Original – beispielsweise in Form eines Scans – eingeblendet werden kann. Dazu existiert in LERA bereits eine prototypische Implementierung für die *Histoire des deux Indes*, wobei die Faksimiles in Seiten aufgeteilt und via der in XML codierten Seitennummern vom System miteinander verknüpft wurden. Dieser Ansatz ist allerdings aufwändig in der Einrichtung sowie durch die Einteilung in Seiten recht ungenau. Eine präzise Zuordnung zwischen den Texten beziehungsweise ihren Segmenten und den konkreten Bildregionen ist wünschenswert und wäre durch bekannte Transkriptionswerkzeuge wie Transkribus [79] oder eScriptorium [82] manuell erstellbar oder möglicherweise mittels Schrifterkennung automatisch möglich.

LERA ist für den Vergleich von Textfassungen eines Werkes ausgelegt. Doch auch zwischen verschiedenen Werken kann es große übereinstimmende Textmengen geben, die aus editionsphilologischer Perspektive interessant sein können. Ein Beispiel hierfür ist das laufende Forschungsprojekt *Aufbau einer digitalen Mehrschicht-Synopse und alphabet-mystische Traditionen zum kabbalistischen Traktat Keter Shem Ṭov* [3], welches das in Abschnitt 2.1.2 beschriebene Editionsprojekt zu *Keter Shem Ṭov* durch die Betrachtung weiterer, in Beziehung stehender Traktate fortführt. Die

anvisierte Mehrschichtsynopse soll die klassische Edition des Werks um eine zusätzliche Dimension erweitern, welche die intertextuellen Bezüge – im konkreten Fall die Wiederverwendung ganzer Abschnitte – erkennt und anschaulich visualisiert. Die entstehenden Lösungen könnten perspektivisch direkt in LERA integriert und so weiteren Editionsvorhaben zugänglich gemacht werden. Diese zusätzliche Dimension würde die bestehende mehrstufige Kollationierung in LERA nochmals erweitern, sodass eine werksübergreifende Exploration möglich wird.

Literaturverzeichnis

[1] Deutsche Forschungsgemeinschaft (DFG). *DFG - GEPRIS - Hannah Arendt. Kritische Gesamtausgabe*. URL: https://gepris.dfg.de/gepris/projekt/421689821 (besucht am 06. 07. 2021).

[2] Deutsche Forschungsgemeinschaft (DFG). *DFG – GEPRIS – Semiautomatische Kollationierung verschiedensprachiger Fassungen eines Textes*. URL: https://gepris.dfg.de/gepris/projekt/524057241 (besucht am 27. 01. 2024).

[3] Deutsche Forschungsgemeinschaft (DFG). *DFG – GEPRIS – Synoptische Edition des kabbalistischen Traktats Keter Shem Tov mit englischer Übersetzung, Stellenkommentar und rezeptionsgeschichtlichen Studien*. URL: https://gepris.dfg.de/gepris/projekt/414786977 (besucht am 11. 06. 2021).

[4] Barbara Aehnlich und Elisabeth Witzenhausen. "LAKomp. Lemmatize, annotate and compare texts in non-standardized languages". In: *RIDE* 15 (2023). Hrsg. von Anna-Maria Sichani und Elena Spadini. DOI: 10.18716/ride.a.15.7.

[5] Akiko Aizawa. "An information-theoretic perspective of tf–idf measures". In: *Information Processing & Management* 39.1 (2003), S. 45–65. DOI: 10.1016/S0306-4573(02)00021-3.

[6] Emily Alder, Jonathan Cranfield, Linda Dryden, Ian Duncan, Christine Ferguson, Simon J. James, Douglas Kerr, Roger Luckhurst, James Machin, Anne Schwan und Jonathan Wild. *Edinburgh Conan Doyle Project*. URL: https://edinburgh-conan-doyle.org (besucht am 14. 07. 2022).

[7] Pat Allan. *Thinking Sphinx on GitHub*. URL: https://github.com/pat/thinking-sphinx (besucht am 27. 01. 2023).

[8] Felipe Almeida und Geraldo Xexéo. “Word embeddings: A survey”. In: *arXiv preprint arXiv:1901.09069* (2019). DOI: 10.48550/arXiv.1901.09069.

[9] Tara L. Andrews und Caroline Macé. “Beyond the tree of texts: Building an empirical model of scribal variation through graph analysis of texts and stemmata”. In: *Literary and Linguistic Computing* 28.4 (2013), S. 504–521. DOI: 10.1093/llc/fqt032.

[10] Bharathi Asokarajan, Ronak Etemadpour, June Abbas, Sam Huskey und Chris Weaver. “TexTile: A Pixel-Based Focus+Context Tool For Analyzing Variants Across Multiple Text Scales”. In: *EuroVis 2017 – Short Papers*. Hrsg. von Barbora Kozlikova, Tobias Schreck und Thomas Wischgoll. The Eurographics Association, 2017. DOI: 10.2312/eurovisshort.20171132.

[11] John F. Atkins und Ray Gesteland. “The 22nd amino acid”. In: *Science* 296.5572 (2002), S. 1409–1410. DOI: 10.1126/science.1073339.

[12] Arturs Backurs und Piotr Indyk. “Edit Distance Cannot Be Computed in Strongly Subquadratic Time (unless SETH is false)”. In: *arXiv e-prints* (2014), arXiv:1412.0348. DOI: 10.48550/arXiv.1412.0348. eprint: 1412.0348.

[13] Nico Balbach, Christian Reul und Frank Puppe. “Typisierte Varianz-Analyse von Texten”. In: *Digital Humanities im deutschsprachigen Raum 2020 – Konferenzabstracts*. Paderborn, 2020, S. 235–238. DOI: 10.5281/zenodo.4621846.

[14] Roman Bleier. “Versioning Machine 5.0 and Audio Versioning”. In: *Proceedings of the 4th International Workshop on Document Changes: Modeling, Detection, Storage and Visualization*. DChanges ’16. Vienna, Austria: Association for Computing Machinery, 2016, S. 1–2. DOI: 10.1145/2993585.2993592.

[15] Mike Bostock. *D3.js – Data-Driven Documents*. URL: https://d3js.org/ (besucht am 16. 06. 2023).

[16] Oliver Bräckel, Hannes Kahl, Friedrich Meins und Charlotte Schubert. “eComparatio – A Software Tool for Automatic Text Comparison”. In: *Digital Classical Philology: Ancient Greek and Latin in the Digital Revolution*. De Gruyter Saur, 2019, S. 221–238. DOI: doi:10.1515/9783110599572-013.

[17] *eComparatio. Editionsvergleich*. Graz, 2015. DOI: 10.5281/zenodo.4623417.

[18] Manuela Bragagnolo. “Les voyages du droit du Portugal à Rome. Le ‘Manual de confessores’ de Martín de Azpilcueta (1492–1586) et ses traductions (The Travels of Law from Portugal to Rome. Martín de Azpilcueta’s ’Manual de confessores’ (1492-1586) and its Translations)”. In: *Max Planck Institute for European Legal History Research Paper Series* 13 (2018). DOI: 10.2139/ssrn.3287684.

[19] Manuela Bragagnolo. *HyperAzpilcueta. Visualizing the instability of early modern normative knowledge - Max-Planck-Institut für Rechtsgeschichte und Rechtstheorie*. URL: https://www.lhlt.mpg.de/forschungsprojekt/hyperazpilcueta (besucht am 22. 06. 2021).

[20] Manuela Bragagnolo. *Martín de Azpilcueta’s Manual for Confessors and the Phenomenon of Epitomisation - Max Planck Institute for Legal History and Legal Theory*. URL: https://www.lhlt.mpg.de/1868502 (besucht am 22. 06. 2021).

[21] Graham Breach. *TagCanvas HTML5 canvas tag cloud on GitHub*. URL: https://github.com/goat1000/TagCanvas (besucht am 20. 01. 2023).

[22] Thomas Bremer, Sylwia Kösser, André Medek, Paul Molitor, Marcus Pöckelmann, Jörg Ritter, Susanne Schütz und Hans-Joachim Solms. *SaDA – Semiautomatische Differenzanalyse komplexer Textvarianten : Schlussbericht des BMBF-Forschungsvorhabens*. Techn. Ber. Förderkennzeichen BMBF 01UG1247. Halle (Saale): Martin-Luther-Universität Halle-Wittenberg, 2015. DOI: 10.2314/GBV:870789287.

[23] Thomas Bremer, Paul Molitor, Marcus Pöckelmann, Jörg Ritter und Susanne Schütz. “Zum Einsatz digitaler Methoden bei der Erstellung und Nutzung genetischer Editionen gedruckter Texte mit verschiedenen Fassungen”.

In: *Editio* 29.1 (2015). Hrsg. von Rüdiger Nutt-Kofoth und Bodo Plachta, S. 29–51. DOI: 10.1515/editio-2015-004.

[24] Peter F Brown, Jennifer C Lai und Robert L Mercer. "Aligning sentences in parallel corpora". In: *29th Annual Meeting of the Association for Computational Linguistics*. 1991, S. 169–176. DOI: 10.3115/981344.981366.

[25] Humberto Carrillo und David Lipman. "The Multiple Sequence Alignment Problem in Biology". In: *SIAM Journal on Applied Mathematics* 48.5 (1988), S. 1073–1082. DOI: 10.1137/0148063.

[26] Stanley F. Chen. "Aligning sentences in bilingual corpora using lexical information". In: *31st Annual Meeting of the Association for Computational Linguistics*. 1993, S. 9–16. DOI: 10.3115/981574.981576.

[27] Ramu Chenna, Hideaki Sugawara, Tadashi Koike, Rodrigo Lopez, Toby J. Gibson, Desmond G. Higgins und Julie D. Thompson. "Multiple sequence alignment with the Clustal series of programs". In: *Nucleic Acids Research* 31.13 (2003), S. 3497–3500. DOI: 10.1093/nar/gkg500.

[28] Biswanath Chowdhury und Gautam Garai. "A review on multiple sequence alignment from the perspective of genetic algorithm". In: *Genomics* 109.5 (2017), S. 419–431. DOI: 10.1016/j.ygeno.2017.06.007.

[29] TEI Consortium. *Text Encoding Initiative*. URL: https://tei-c.org/ (besucht am 03. 12. 2023).

[30] Oracle Corporation. *MySQL*. URL: https://www.mysql.com/ (besucht am 27. 01. 2024).

[31] James Cummings und Arno Mittelbach. "The Holinshed Project: Comparing and linking two editions of Holinshed's Chronicle". In: *International Journal of Humanities and Arts Computing* 4.1-2 (2010), S. 39–53. DOI: 10.3366/ijhac.2011.0006.

[32] Janis Dähne, Marcus Pöckelmann und Jörg Ritter. "Word Clouds with Spatial Stable Word Positions across Multiple Text Witnesses". In: *Digital Humanities 2023: Book of Abstracts*. Graz, 2023, S. 429–431. DOI: 10.5281/zenodo.8107985.

[33] Janis Dähne, Marcus Pöckelmann, Jörg Ritter und Paul Molitor. "Putting collation of text witnesses on a formal basis". In: *Digital Scholarship in the Humanities* 37.2 (2021), S. 375–390. DOI: 10.1093/llc/fqab058.

[34] Fred J. Damerau. “A Technique for Computer Detection and Correction of Spelling Errors”. In: *Commun. ACM* 7.3 (1964), S. 171–176. DOI: 10.1145/363958.363994.

[35] Margaret Dayhoff, R. Schwartz und B. Orcutt. “A Model of Evolutionary Change in Proteins”. In: *Atlas of protein sequence and structure* 5 (1978), S. 345–352.

[36] Ronald H. Dekker, Dirk van Hulle, Gregor Middell, Vincent Neyt und Joris van Zundert. “Computer-supported collation of modern manuscripts: CollateX and the Beckett Digital Manuscript Project”. In: *Digital Scholarship in the Humanities* 30.3 (2015), S. 452–470.

[37] Ronald H. Dekker und Gregor Middell. “Computer-supported collation with CollateX: managing textual variance in an environment with varying requirements”. In: *Supporting Digital Humanities* 2 (2011).

[38] Ronald H. Dekker und Gregor Middell. *CollateX – Software for Collating Textual Sources*. URL: https://collatex.net (besucht am 27. 04. 2021).

[39] Jacob Devlin, Ming-Wei Chang, Kenton Lee und Kristina Toutanova. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018). DOI: 10.48550/arXiv.1810.04805.

[40] Kevin S. Dias. *Pragmatic Segmenter on GitHub*. URL: https://github.com/diasks2/pragmatic_segmenter (besucht am 07. 03. 2021).

[41] e-editiones.org. *TEI Publisher – The Instant Publishing Toolbox*. URL: https://teipublisher.com (besucht am 11. 08. 2023).

[42] Sean R. Eddy. “Multiple alignment using hidden Markov models”. In: *Ismb*. Bd. 3. 1995, S. 114–120.

[43] Oualid El Khattabi, Dima M. Sakran, Agnes Kloocke, Ruslan Pavlyshyn, Marwa M. Ahmed, Rima Redwan, Victoria Mummelthei, Mahmoud Kozae, Johannes Stephan, Khouloud Khalfallah, Theodore S. Beers, Isabel Toral und Beatrice Gründler. *Kalīla and Dimna – AnonymClassic*. URL: https://www.geschkult.fu-berlin.de/en/e/kalila-wa-dimna (besucht am 16. 06. 2021).

[44] Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan und Wei Wang. “Language-agnostic BERT sentence embedding”. In: *arXiv preprint arXiv:2007.01852* (2020). DOI: 10.48550/arXiv.2007.01852.

[45] Da-Fei Feng und Russell F. Doolittle. “Progressive sequence alignment as a prerequisitetto correct phylogenetic trees”. In: *Journal of molecular evolution* 25 (1987), S. 351–360. DOI: 10.1007/BF02603120.

[46] Deutsche Forschungsgemeinschaft. “Scientific Library Services and Information Systems (LIS): DFG Practical Guidelines on Digitisation”. In: (2009).

[47] OpenJS Foundation. *jQuery*. URL: https://jquery.com/ (besucht am 27. 01. 2024).

[48] Greta Franzini, Simon Mahony und Melissa Terras. “A catalogue of digital editions”. In: *Digital Scholarly Editing: Theories and Practices*. Hrsg. von E Pierazzo und M Driscoll. Cambridge: Open Book Publishers, 2016, S. 161–182. DOI: 10.11647/OBP.0095.

[49] Ben Fry. *On the Origin of Species: The Preservation of Favoured Traces*. 2009. URL: https://benfry.com/traces/ (besucht am 12. 11. 2023).

[50] Alejandro García-Reidy. “El cortesano embustero, una supuesta comedia olvidada de Lope de Vega, y La española, de Cepeda”. In: *Revista de Filología Española* 102.2 (2022), S. 407–432. DOI: 10.3989/rfe.2022.015.

[51] Michael R. Garey und David S. Johnson. *Computers and intractability*. New York: freeman San Francisco, 1979.

[52] André Gießler, Jörg Ritter, Paul Molitor, Martin Andert, Sylwia Kösser und Aletta Leipold. “User-friendly lemmatization and morphological annotation of Early New High German manuscripts”. In: *Digital Humanities 2014 – Book of Abstracts*. Lausanne, 2014, S. 469–471.

[53] Beatrice Gründler. “The Arabic Anonymous in a World Classic (Acronym: AnonymClassic)”. In: *Geschichte der Germanistik* 51/52 (2017), S. 156–157.

[54] Beatrice Gründler, Jan J van Ginkel, Rima Redwan, Khouloud Khalfallah, Isabel Toral, Johannes Stephan, Matthew L Keegan, Theodore S Beers, Mahmoud Kozae und Marwa M Ahmed. “An Interim Report on the Editorial and Analytical Work of the AnonymClassic Project”. In: *medieval worlds* 11 (2020), S. 241–279. DOI: 10.1553/medievalworlds_no11_2020s241.

[55] Beatrice Gründler und Marcus Pöckelmann. “Adjusting LERA for the Comparison of Arabic Manuscripts of Kalīla wa-Dimna”. In: *Digital Humanities 2018 – Book of Abstracts*. Mexico City, 2018, S. 467–468.

[56] Dan Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Bd. 11. New York: Cambridge University Press, 1997.

[57] Susanne Haaf, Frank Wiegand und Alexander Geyken. “Measuring the correctness of double-keying: Error classification and quality control in a large corpus of TEI-annotated historical text”. In: *Journal of the Text Encoding Initiative* 4 (2013). DOI: 10.4000/jtei.739.

[58] Barbara Hahn, Anne Eusterschulte, Ingo Kieslich und Christian Pische. *Hannah Arendt Digital*. URL: https://hannah-arendt-edition.net (besucht am 06. 07. 2021).

[59] Jiyeon Ham und Eun-Sol Kim. “Semantic Alignment with Calibrated Similarity for Multilingual Sentence Embedding”. In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Hrsg. von Marie-Francine Moens, Xuanjing Huang, Lucia Specia und Scott Wen-tau Yih. Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021, S. 1781–1791. DOI: 10.18653/v1/2021.findings-emnlp.153.

[60] Waqar Haque, Alex Aravind und Bharath Reddy. “Pairwise Sequence Alignment Algorithms: A Survey”. In: *Proceedings of the 2009 Conference on Information Science, Technology and Applications*. ISTA ’09. Kuwait, Kuwait: Association for Computing Machinery, 2009, S. 96–103. DOI: 10.1145/1551950.1551980.

[61] David Heinemeier Hansson, Darragh Curran, Neha Batra, Duncan Davidson, Steve Davis, Rosa Gutiérrez, Bruno Miranda, Jorge Valdivia und Miles Woodroffe. *Ruby on Rails – A web-app framework that includes everything*

needed to create database-backed web applications according to the Model-View-Controller (MVC) pattern. URL: https://rubyonrails.org (besucht am 27. 01. 2024).

[62] Steven Henikoff und Jorja G. Henikoff. "Amino Acid Substitution Matrices from Protein Blocks". In: *Proceedings of the National Academy of Sciences* 89.22 (1992), S. 10915–10919. DOI: 10.1073/pnas.89.22.1091.

[63] Elisa Herold, Marcus Pöckelmann, Christian Berg, Jörg Ritter und Mark M. Hall. "Stable Word-Clouds for Visualising Text-Changes Over Time". In: *Digital Libraries for Open Knowledge*. Hrsg. von Antoine Doucet, Antoine Isaac, Koraljka Golub, Trond Aalberg und Adam Jatowt. Springer International Publishing, 2019, S. 224–237. DOI: 10.1007/978-3-030-30760-8_20.

[64] Armin Hoenen. "Das erste dynamische Stemma, Pionier des digitalen Zeitalters?" In: *Digital Humanities im deutschsprachigen Raum 2016 – Konferenzabstracts*. Leipzig, 2016, S. 359–361. DOI: 10.5281/zenodo.4645172.

[65] Paulien Hogeweg und Ben Hesper. "The alignment of sets of sequences and the construction of phyletic trees: an integrated method". In: *Journal of molecular evolution* 20 (1984), S. 175–186. DOI: 10.1007/BF02257378.

[66] Jörg Hörnschemeyer. "Textgenetische Prozesse in Digitalen Editionen". Diss. Universität zu Köln, 2017.

[67] Amin Hosseininasab und Willem-Jan Van Hoeve. "Exact multiple sequence alignment by synchronized decision diagrams". In: *INFORMS Journal on Computing* 33.2 (2021), S. 721–738. DOI: 10.1287/ijoc.2019.0937.

[68] Christopher J. Howe, Adrian C. Barbrook, Matthew Spencer, Peter Robinson, Barbara Bordalejo und Linne R. Mooney. "Manuscript evolution". In: *Trends in Genetics* 17.3 (2001), S. 147–152. DOI: 10.1016/S0168-9525(00)02210-1.

[69] James Wayne Hunt und M. Douglas MacIlroy. *An algorithm for differential file comparison*. Techn. Ber. 41. Computing Science Technical Report. Murray Hill: Bell Laboratories, 1976.

[70] Heikki Hyyrö. “A bit-vector algorithm for computing Levenshtein and Damerau edit distances”. In: *Nordic Journal of Computing* 10.1 (2003), S. 29–39.

[71] Heikki Hyyrö. “Practical methods for approximate string matching”. Diss. University of Tampere, 2003.

[72] IntereditionWiki. *About microservices*. URL: http://www.interedition.eu/wiki/index.php/About_microservices (besucht am 27. 04. 2021).

[73] Paul Jaccard. “Étude comparative de la distribution florale dans une portion des Alpes et des Jura”. In: *Bulletin de la Société Vaudoise des Sciences Naturelles* 37 (1901), S. 547–579. DOI: 10.5169/seals-266450.

[74] Stefan Jänicke, Annette Geßner, Marco Büchler und Gerik Scheuermann. “Visualizations for Text Re-use”. In: *2014 International Conference on Information Visualization Theory and Applications (IVAPP)*. IEEE. 2014, S. 59–70.

[75] Stefan Jänicke, Annette Geßner, Greta Franzini, Melissa Terras, Simon Mahony und Gerik Scheuermann. “TRAViz: A Visualization for Variant Graphs”. In: *Digital Scholarship in the Humanities* 30.suppl_1 (2015), S. i83–i99. DOI: 10.1093/llc/fqv049.

[76] Stefan Jänicke und David Joseph Wrisley. “Interactive visual alignment of medieval text versions”. In: *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*. IEEE. 2017, S. 127–138.

[77] Fotis Jannidis, Hubertus Kohle und Malte Rehbein, Hrsg. *Digital Humanities: Eine Einführung*. 1. Aufl. Stuttgart: J.B. Metzler Verlag, 2017. DOI: 10.1007/978-3-476-05446-3.

[78] Steven Swann Jones. “On Analyzing Fairy Tales:”Little Red Riding Hood”Revisited”. In: *Western Folklore* 46.2 (1987), S. 97–106. DOI: 10.2307/1499927.

[79] Philip Kahle, Sebastian Colutto, Günter Hackl und Günter Mühlberger. “Transkribus-a service platform for transcription, recognition and retrieval of historical documents”. In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Bd. 4. IEEE. 2017, S. 19–24.

[80] Erik Ketzan und Christof Schöch. “Classifying and Contextualizing Edits in Variants with Coleto: Three Versions of Andy Weir’s The Martian.” In: *DHQ: Digital Humanities Quarterly* 15.4 (2021).

[81] Ingo Kieslich und Christian Pische. *Hannah Arendt. Kritische Gesamtausgabe*. URL: https://www.arendteditionprojekt.de (besucht am 06. 07. 2021).

[82] Benjamin Kiessling, Robin Tissot, Peter Stokes und Daniel Stökl Ben Ezra. “eScriptorium: an open source platform for historical document analysis”. In: *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*. Bd. 2. IEEE. 2019, S. 19–19.

[83] Matthew J. Kusner, Y Sun, N. I Kolkin und K. Q. Weinberger. “From Word Embeddings to Document Distances”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Lille, France, 2015, S. 957–966.

[84] Aletta Leipold, Sylwia Kösser, André Gießler und Hans-Joachim Solms. “Zwischen Online-Korpus und Buch. Die Hybridedition der Wundarznei des Heinrich von Pfalzpaint”. In: (2015). Hrsg. von Thomas Bein, S. 167–184. DOI: 10.1515/9783110418255-014.

[85] Vladimir I. Levenshtein. “Binary Codes Capable of Correcting Deletions, Insertions, and Reversals”. In: *Soviet Physics Doklady* 10.8 (1966), S. 707–710.

[86] Ruiqi Li, Xiang Zhao und Marie-Francine Moens. “A Brief Overview of Universal Sentence Representation Methods: A Linguistic View”. In: *ACM Comput. Surv.* 55.3 (2022). DOI: 10.1145/3482853.

[87] Steffen Lohmann, Jürgen Ziegler und Lena Tetzlaff. “Comparison of Tag Cloud Layouts: Task-Related Performance and Visual Exploration”. In: *Human-Computer Interaction – INTERACT 2009*. Hrsg. von Tom Gross, Jan Gulliksen, Paula Kotzé, Lars Oestreicher, Philippe Palanque, Raquel Oliveira Prates und Marco Winckler. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, S. 392–404. DOI: 10.1007/978-3-642-03655-2_43.

[88] Jan Luhmann und Manuel Burghardt. “Digital humanities—A discipline in its own right? An analysis of the role and position of digital humanities in the academic landscape”. In: *Journal of the Association for Information Science and Technology* 73.2 (2022), S. 148–171. DOI: 10.1002/asi.24533.

[89] Thi Kim Hanh Luu, Marcus Pöckelmann, Jörg Ritter und Paul Molitor. “Applying LERA for collating witnesses of The Tale of Kiều, a Vietnamese poem written in Nôm script”. In: *Digital Humanities 2022 – Book of Abstracts*. Tokyo, 2022, S. 302–305.

[90] Luis Marujo, Márcio Viveiros und João Paulo da Silva Neto. “Keyphrase cloud generation of broadcast news”. In: *arXiv preprint arXiv:1306.4606* (2013). DOI: 10.48550/arXiv.1306.4606.

[91] Bruce M. Metzger. *The Bible in translation: Ancient and English versions*. Baker Academic, 2001.

[92] Gregor Middell. *Textual Variance – TEIWiki*. URL: https://wiki.tei-c.org/index.php/Textual_Variance (besucht am 27. 04. 2021).

[93] George A. Miller. “WordNet: A Lexical Database for English”. In: *Commun. ACM* 38.11 (1995), S. 39–41. DOI: 10.1145/219717.219748.

[94] Paul Molitor, Gerold Necker, Marcus Pöckelmann, Bill Rebiger und Jörg Ritter. “Keter Shem Ṭov – Prozessualisierung eines Editionsprojekts mit 100 Textzeugen”. In: *DHd2020, 7. Jahrestagung der Digital Humanities im deutschsprachigen Raum*. Paderborn, 2020. DOI: 10.5281/zenodo.4621883.

[95] Paul Molitor, Marcus Pöckelmann, André Medek und Jörg Ritter. *SaDA – Semi-automatische Differenzanalyse von komplexen Textvarianten*. URL: https://sada.uzi.uni-halle.de (besucht am 26. 05. 2021).

[96] Michio Murata, Jane S Richardson und Joel L Sussman. “Simultaneous comparison of three protein sequences”. In: *Proceedings of the National Academy of Sciences* 82.10 (1985), S. 3073–3077. DOI: 10.1073/pnas.82.10.3073.

[97] Eugene W. Myers. “An O (ND) difference algorithm and its variations”. In: *Algorithmica* 1.1-4 (1986), S. 251–266. DOI: 10.1007/BF01840446.

[98] Mohamadou Nassourou. “Design and Implementation of Architectures for Interactive Textual Documents Collation Systems”. Preprint unter: urn:nbn:de:bvb:20-opus-56601. 2011.

[99] Gerold Necker, Paul Molitor, Bill Rebiger, Marcus Pöckelmann und Maximilian de Molière. *Kabbalah Editions*. URL: https://kabbalaheditions.org (besucht am 11. 06. 2021).

[100] Gerold Necker und Bill Rebiger. “The One and the Many: The Structure and Versions of Keter Shem Tov According to the New Synoptic Edition with a Special Focus on MS Jerusalem, NLI, 8°541”. In: *Proceedings of the Workshop Editing Kabbalistic Texts*. Erscheint 2024. Harrassowitz.

[101] Saul B. Needleman und Christian D. Wunsch. “A general method applicable to the search for similarities in the amino acid sequence of two proteins”. In: *Journal of Molecular Biology* 48.3 (1970), S. 443–453. DOI: 10.1016/0022-2836(70)90057-4.

[102] NINES. *Juxta 1.6 User's Manual*. 2011. URL: http://www.juxtasoftware.org/images/wiki/JuxtaManual_1.6.pdf (besucht am 25. 01. 2022).

[103] NINES. *Juxta Commons*. 2013. URL: http://www.juxtacommons.org (besucht am 23. 01. 2020).

[104] Cédric Notredame, Desmond G. Higgins und Jaap Heringa. “T-coffee: a novel method for fast and accurate multiple sequence alignment11Edited by J. Thornton”. In: *Journal of Molecular Biology* 302.1 (2000), S. 205–217. DOI: 10.1006/jmbi.2000.4042.

[105] Paulo A. S. Nuin, Zhouzhi Wang und Elisabeth RM Tillier. “The accuracy of several multiple sequence alignment programs for proteins”. In: *BMC bioinformatics* 7.1 (2006), S. 1–18. DOI: 10.1186/1471-2105-7-471.

[106] Elisa Nury. “Automated Collation and Digital Editions – From Theory to Practice”. Diss. King's College London, 2018.

[107] Elisa Nury. “Visualizing collation results”. In: *Variants. The Journal of the European Society for Textual Scholarship* 14 (2019), S. 75–94.

[108] Elisa Nury und Elena Spadini. “From Giant Despair to a New Heaven: The Early Years of Automatic Collation”. In: *it – Information Technology* 62.2 (2020).

[109] Wilhelm Ott. “Strategies and tools for textual scholarship: the Tübingen system of text processing programs (TUSTEP)”. In: *Literary and Linguistic Computing* 15.1 (2000), S. 93–108. DOI: 10.1093/llc/15.1.93.

[110] Wilhelm Ott und Tobias Ott. “TXSTEP – an integrated XML-based scripting language for scholarly text data processing”. In: *Conference Proceedings of XML Prague 2015* (2015), S. 191–195.

[111] David C. Parker. “The text of the New Testament and computers: The International Greek New Testament Project”. In: *Literary and Linguistic Computing* 15.1 (2000), S. 27–41. DOI: 10.1093/llc/15.1.27.

[112] Aaron Patterson, Mike Dalessio, Charles Nutter, Sergio Arbeo, Patrick Mahoney, Yoko Harada, Akinori Musha, John Shahid und Lars Kanis. *Nokogiri-Homepage*. URL: https://nokogiri.org (besucht am 20. 02. 2021).

[113] Slav Petrov, Dipanjan Das und Ryan McDonald. “A universal part-of-speech tagset”. In: *arXiv preprint arXiv:1104.2086* (2011).

[114] Elena Pierazzo. “What future for digital scholarly editions? From Haute Couture to Prêt-à-Porter”. In: *International Journal of Digital Humanities* 1 (2019), S. 209–220. DOI: 10.1007/s42803-019-00019-3.

[115] Marcus Pöckelmann. “Extensions of the Digital Collation Tool LERA for the Scholarly Edition of Keter Shem Ṭov”. In: *Proceedings of the Workshop Editing Kabbalistic Texts*. Erscheint 2024. Harrassowitz.

[116] Marcus Pöckelmann, Janis Dähne, Jörg Ritter und Paul Molitor. “Fast paraphrase extraction in Ancient Greek literature”. In: *it – Information Technology* 62.2 (2020), S. 75–89. DOI: 10.1515/itit-2019-0042.

[117] Marcus Pöckelmann, André Medek, Paul Molitor und Jörg Ritter. “CATview-Supporting the Investigation of Text Genesis of Large Manuscripts by an Overall Interactive Visualization Tool”. In: *Digital Humanities 2015 – Abstracts*. Sydney, 2015.

[118] Marcus Pöckelmann, André Medek, Jörg Ritter und Paul Molitor. “LERA – An interactive platform for synoptical representations of multiple text witnesses”. In: *Digital Scholarship in the Humanities* (2022). DOI: 10.1093/llc/fqac021.

[119] Marcus Pöckelmann, Paul Molitor und Jörg Ritter. *CATview – the Colored & Aligned Texts view*. URL: https://catview.uzi.uni-halle.de (besucht am 15. 01. 2021).

[120] Marcus Pöckelmann, Jörg Ritter und Paul Molitor. *LERA – Locate, Explore, Retrace and Apprehend complex text variants*. URL: https://lera.uzi.uni-halle.de (besucht am 31. 01. 2022).

[121] Marcus Pöckelmann, Julia Ritter und André Gießler. “On Automatically Disambiguating End-of-line Hyphenated Words in French Texts”. In: *Digital Humanities 2014 – Book of Abstracts*. Lausanne, 2014, S. 313–315.

[122] Kenneth M. Price. “Electronic Scholarly Editions”. In: *A Companion to Digital Literary Studies*. Hrsg. von Susan Schreibman und Ray Siemens. John Wiley & Sons, 2013. Kap. 24. DOI: 10.1002/9781405177504.ch24.

[123] Guillaume-Thomas Raynal. *Histoire philosophique et politique des établissements et du commerce des Européens dans les deux Indes. Édition critique*. Hrsg. von Anthony Strugnell. Bd. 1. Centre International d’Etude du XVIIIe siècle, 2010.

[124] Guillaume-Thomas Raynal. *Histoire philosophique et politique des établissements et du commerce des Européens dans les deux Indes. Édition critique*. Hrsg. von Andrew Brown und Hans-Jürgen Lüsebrink. Bd. 2. Centre International d’Etude du XVIIIe siècle, 2018.

[125] Guillaume-Thomas Raynal. *Histoire philosophique et politique des établissements et du commerce des Européens dans les deux Indes. Édition critique*. Hrsg. von Cecil Courtney. Bd. 3. Centre International d’Etude du XVIIIe siècle, 2020.

[126] Community Research und Development Information Service (CORDIS). *The Arabic Anonymous in a World Classic - AnonymClassic Project - H2020 - CORDIS - European Commission*. URL: https://cordis.europa.eu/project/id/742635 (besucht am 16. 06. 2021).

[127] Stephen Robertson. “Understanding inverse document frequency: on theoretical arguments for IDF”. In: *Journal of documentation* 60.5 (2004), S. 503–520. DOI: 10.1108/00220410410560582.

[128] Peter Robinson. “The Collation and Textual Criticism of Icelandic Manuscripts (1): Collation”. In: *Literary and Linguistic Computing* 4.2 (1989), S. 99–105. DOI: 10.1093/llc/4.2.99.

[129] Torsten Roeder. “Review of Juxta Web Service, LERA, and Variance Viewer. Web based collation tools for TEI”. In: *RIDE* 11 (2020). Hrsg. von Anna-Maria Sichani und Elena Spadini. DOI: 10.18716/ride.a.11.5.

[130] Teemu Roos und Yuan Zou. “Analysis of Textual Variation by Latent Tree Structures”. In: *2011 IEEE 11th International Conference on Data Mining*. 2011, S. 567–576. DOI: 10.1109/ICDM.2011.24.

[131] Sebastian Ruder, Ivan Vulić und Anders Søgaard. “A survey of cross-lingual word embedding models”. In: *Journal of Artificial Intelligence Research* 65 (2019), S. 569–631. DOI: 10.1613/jair.1.11640.

[132] N. Saitou und M. Nei. “The neighbor-joining method: a new method for reconstructing phylogenetic trees.” In: *Molecular Biology and Evolution* 4.4 (1987), S. 406–425. DOI: 10.1093/oxfordjournals.molbev.a040454.

[133] Kuno Schälkle und Wilhelm Ott. *TUSTEP – Handbuch und Referenz (Version 2022)*. URL: https://www.tustep.uni-tuebingen.de/pdf/handbuch.pdf (besucht am 24. 01. 2022).

[134] Helmut Schmid. “Probabilistic Part-of-Speech Tagging Using Decision Trees”. In: *Proceedings of International Conference on New Methods in Language Processing*. Manchester, UK, 1994, S. 44–49.

[135] Desmond Schmidt. “Merging Multi-Version Texts: a Generic Solution to the Overlap Problem”. In: *Proceedings of Balisage: The Markup Conference*. Bd. 3. Montréal, 2009. DOI: 10.4242/BalisageVol3.Schmidt01.

[136] Desmond Schmidt. “Collation on the Web”. In: *Proceedings of the Digital Humanities 2013*. University of Nebraska-Lincoln, 2013, S. 378–380.

[137] Desmond Schmidt und Robert Colomb. “A data structure for representing multi-version texts online”. In: *International Journal of Human-Computer Studies* 67.6 (2009), S. 497–514. DOI: 10.1016/j.ijhcs.2009.02.001.

[138] Susan Schreibman, Amit Kumar und Jarom McDonald. “The versioning machine”. In: *Literary and Linguistic Computing* 18.1 (2003), S. 101–107.

[139] Susanne Schütz und Marcus Pöckelmann. “LERA – Explorative Analyse komplexer Textvarianten in Editionsphilologie und Diskursanalyse”. In: *Digital Humanities im deutschsprachigen Raum 2016 – Konferenzabstracts*. Leipzig, 2016, S. 249–253. DOI: 10.5281/zenodo.4645364.

[140] Christin Seifert, Barbara Kump, Wolfgang Kienreich, Gisela Granitzer und Michael Granitzer. “On the Beauty and Usability of Tag Clouds”. In: *2008 12th International Conference Information Visualisation*. 2008, S. 17–25. DOI: 10.1109/IV.2008.89.

[141] Temple F. Smith und Michael S. Waterman. “Identification of common molecular subsequences”. In: *Journal of Molecular Biology* 147.1 (1981), S. 195–197. DOI: 10.1016/0022-2836(81)90087-5.

[142] Karen Spärck Jones. “A statistical interpretation of term specificity and its application in retrieval”. In: *Journal of documentation* 28.1 (1972), S. 11–21. DOI: doi.org/10.1108/eb026526.

[143] Matthew Spencer und Christopher J. Howe. “Collating Texts Using Progressive Multiple Alignment”. In: *Computers and the Humanities* 38.3 (2004), S. 253–270.

[144] Theodor Sperlea. *Multiple Sequenzalignments*. Springer, 2019. DOI: 10.1007/978-3-662-58811-6.

[145] Klaus Thoden, Juliane Stiller, Natasa Bulatovic, Hanna-Lena Meiners und Nadia Boukhelifa. “User-Centered Design Practices in Digital Humanities – Experiences from DARIAH and CENDARI”. In: *ABI Technik* 37.1 (2017), S. 2–11. DOI: 10.1515/abitech-2017-0002.

[146] Esko Ukkonen. “Algorithms for approximate string matching”. In: *Information and Control* 64.1-3 (1985), S. 100–118. DOI: 10.1016/S0019-9958(85)80046-2.

[147] J Craig Venter, Mark D Adams, Eugene W Myers, Peter W Li, Richard J Mural, Granger G Sutton, Hamilton O Smith, Mark Yandell, Cheryl A Evans, Robert A Holt und weitere. “The Sequence of the Human Genome”. In: *Science* 291.5507 (2001), S. 1304–1351. DOI: 10.1126/science.1058040.

[148] Andreas Wagner und Manuela Bragagnolo. “Multimodale Versuche der Alignierung historischer Texte”. In: *DHd2019, 6. Jahrestagung der Digital Humanities im deutschsprachigen Raum*. Frankfurt, 2019, S. 181–184. DOI: 10.5281/zenodo.4622224.

[149] Robert A Wagner. “On the complexity of the extended string-to-string correction problem”. In: *Proceedings of the seventh annual ACM symposium on theory of computing*. 1975, S. 218–223. DOI: 10.1145/800116.803771.

[150] Robert A. Wagner und Michael J. Fischer. “The String-to-String Correction Problem”. In: *J. ACM* 21.1 (1974), S. 168–173. DOI: 10.1145/321796.321811.

[151] Robert A. Wagner und Roy Lowrance. “An Extension of the String-to-String Correction Problem”. In: *J. ACM* 22.2 (1975), S. 177–183. DOI: 10.1145/321879.321880.

[152] Lusheng Wang und Tao Jiang. “On the complexity of multiple sequence alignment”. In: *Journal of computational biology* 1.4 (1994), S. 337–348. DOI: 10.1089/cmb.1994.1.337.

[153] Dana Wheeles und Jensen Kristin. “Juxta Commons”. In: *Proceedings of the Digital Humanities 2013*. University of Nebraska-Lincoln, 2013, S. 545–546.

[154] Ken Whistler. *Unicode Normalization Forms*. URL: https://unicode.org/reports/tr15 (besucht am 15. 01. 2021).

[155] Ken Whistler und Laurențiu Iancu. *Unicode Character Database - General Category Values*. URL: https://www.unicode.org/reports/tr44/#General_Category_Values (besucht am 10. 03. 2021).

[156] Tariq Yousef und Stefan Janicke. “A Survey of Text Alignment Visualization”. In: *IEEE Transactions on Visualization and Computer Graphics* 27.2 (2021), S. 1149–1159. DOI: 10.1109/TVCG.2020.3028975.

[157] Andrejka Žejn. *Izhodišča slovenske pripovedne proze*. URL: https://ispp.zrc-sazu.si/ (besucht am 28. 09. 2023).

Anhang A

Laufzeiten bei umfangreichen Textfassungen

Im Folgenden werden die für einige Kollationierungswerkzeuge in Abschnitt 3.2.2 genannten Laufzeiten bei der Kollationierung umfangreicher Textfassungen detailliert aufgeschlüsselt sowie die verwendeten Materialien und Methoden dargelegt.
Der Fokus der durchgeführten Versuche lag dabei rein auf den benötigten Laufzeiten der verschiedenen Werkzeuge für unterschiedliche Eingabegrößen. Damit wird im Rahmen der Dissertation die Frage beantwortet, inwieweit umfangreiche Textfassungen mit den Werkzeugen überhaupt kollationiert werden können. Die Qualität der Ergebnisse war dabei kein Gegenstand der Untersuchung.

Materialien und Methoden

Datensatz

Für das Experiment wurde das sechste Buch der *Histoire des deux Indes* in zwei beziehungsweise vier Textfassungen genutzt (siehe Abschnitt 2.1.1). Bei den Versuchen mit zwei Textfassungen wurden dabei die beiden längsten Textfassungen (*H80* und *H20*) verwendet.
Die Texte wurden normalisiert. So handelt es bei der Datengrundlage um fortlaufenden Fließtext, der durch Leerzeichen getrennte Wörter enthält. Hinzu kommen Zeilenumbrüche, welche die Absatzgrenzen des ursprünglichen Textes repräsentieren. Alle übrigen Satzzeichen wurden entfernt, was auch das kaufmännische Und

(&) mit einschließt. Zudem wurde die Zeilentrennung von Wörtern aufgelöst (siehe Abschnitt 4.2.2) sowie Vorkommen des langen s normalisiert (iſt → ist). Die originale Groß-/Kleinschreibung und Verwendung von Diakritika blieben unverändert.
Aus dieser normalisierten Datengrundlage wurden Datensätze für zwei und vier Textfassungen generiert, welche Auszüge der Datengrundlage enthalten. Die unterschiedlichen Datensätze werden im Folgenden schematisch als x_y bezeichnet, wobei die Variable x die Anzahl der Textfassungen und die Variable y die Anzahl der Wörter pro Textfassung anzeigt. Zunächst wurden dabei Auszüge mit den je ersten 500 Wörtern extrahiert (Datensätze 2_500 und 4_500). Diese Textmenge wird dann sukzessive verdoppelt. Dabei weisen die Textfassungen *H70*, *H74*, *H80* und *H20* in ihrer normalisierten Form einen Umfang von 26.716, 28.671, 41.285 und 50.217 Wörtern auf. Demnach konnten für zwei Textfassungen (*H80* und *H20*) Datensätze für bis zu 32.000 Wörter, für alle vier Textfassungen bis zu 16.000 Wörter angelegt werden. Hinzu kommen die zwei Datensätze 2_all und 4_all, welche die Textfassungen in ihrer Gesamtlänge beinhalten.

Hardware

Die untersuchten Werkzeuge wurden auf der selben Hardware getestet. Dabei handelt es sich um einen Laptop mit einem 64-bit-Betriebssystem (Ubuntu 22.04), vier Kernen mit einer Taktung von je 2,40GHz (Intel® Core™ i7-5500U) und acht GiB DDR3-Arbeitsspeicher mit einer Geschwindigkeit von 1600 MT/s. Für die Untersuchung der Werkzeuge TRAViz und eComparatio war ein Browser notwendig, wofür Mozilla Firefox in der Version 121.0 verwendet wurde.

Werkzeuge und Laufzeitmessung

Die untersuchten und unten aufgeführten Laufzeiten entsprechen der benötigten Wall-Clock-Time der Werkzeuge. Dabei handelt es sich um die zeitliche Differenz zwischen dem Start und der Fertigstellung einer Berechnung und damit um die tatsächliche Wartezeit für Nutzer:innen bei Verwendung der Werkzeuge. Die Experimente besitzen damit eine anwendungsorientierte Perspektive. Bei der Wall-Clock-Time wird die Laufzeit der Werkzeuge allerdings auch von der generellen Auslastung des Computers, genauer der Verwendung der verfügbaren Ressourcen durch andere Programme, beeinflusst. Um diesen Einfluss möglichst gering und fair

zu halten, wurden während der Experimente keine nicht zwingend benötigten Programme ausgeführt und der Laptop stets im Netzbetrieb verwendet.

In den Experimenten wurden die vier Werkzeuge TRAViz, CollateX, eComparatio und LERA untersucht, indem mit diesen sukzessive größer werdende Datensätze kollationiert wurden. Dabei wurden, wenn nicht anders genannt, die Standardeinstellungen der Werkzeuge beibehalten. Zudem wurden die Versuchsreihen abgebrochen, wenn die Laufzeit die Grenze von einer Stunde überschritten oder die Eingabegröße zu einem Absturz des Werkzeugs geführt hat.

Anmerkungen zu TRAViz

Für den Test von TRAViz [75] wurde eine rudimentäre Webapplikation angelegt, welche das Werkzeug in der neusten Version (8. April 2016) aus dem Projekt-Repositorium[1] sowie die Datensätze enthält. Zudem wurde eine Zeitmessung auf Basis der aktuellen Systemzeit integriert. Dabei können die von TRAViz implementierten Funktionen der Alignierung und Visualisierung getrennt voneinander betrachtet werden.

Anmerkungen zu CollateX

Für die Untersuchung von CollateX [37] wurde die aktuelle Version 1.7.1 (17. Dezember 2015) aus dem Projekt-Repositorium[2] verwendet. Die Software wurde als Stand-Alone-Werkzeug installiert und mittels Kommandozeilenaufruf getestet, wobei Versuche mit allen drei integrierten Vergleichsalgorithmen (Dekker, Needleman-Wunsch, MEDITE) separat mittels entsprechender Parameterwahl durchgeführt wurden. Die Datensätze wurden dazu als JSON-Datei aufbereitet und übergeben, die Ergebnisse der Kollationierung wiederum als XML-Datei gespeichert. Zur Laufzeitmessung wurde der *time*-Befehl von Linux dem Aufruf von CollateX vorangestellt, welcher unter anderem die „real-time“ und damit die gewünschte Wall-Clock-Time bei der Ausführung des Befehls misst.

[1] Unter: `https://github.com/stjaenicke/TRAViz` (besucht am 13.07.2022).

[2] Unter: `https://github.com/interedition/collatex` (besucht am 22.05.2022).

Anmerkungen zu eComparatio

Das Werkzeug eComparatio [16] wurde ebenfalls in der neusten Version (3. Februar 2021) getestet, welche aus dem Projekt-Repositorium[3] heruntergeladen und lokal installiert wurde. Zur Nutzung des webbasierten Werkzeugs musste ein Python-Server gestartet werden, da der in den Experimenten verwendete Browser (Firefox in Version 121.0) ein direktes Ausführen aus Sicherheitsgründen verbietet[4]. Darüber hinaus enthielt die getestete Version von eComparatio einige Fehler im Javascript-Quellcode, welche zunächst behoben werden mussten[5]. Einige verbleibende, im Debbuging-Tool des Browsers angezeigte Fehlermeldungen wurden für das Experiment ignoriert, da sie keine Auswirkungen auf die eigentliche Ausführung von eComparatio zu haben schienen. Um Experimente mit allen Datensätzen durchführen zu können, musste zudem der durch den Browser zulässige lokale Speicher für eComparatio erhöht werden. Schließlich wurde auch eine Zeitmessung direkt im Quellcode ergänzt, welche beide, von eComparatio intern aufgerufene Vergleichsfunktionen („ecomparatioVerg“ und „metavergleich“) umschließt. Die Datensätze wurden anschließend mittels der grafischen Nutzeroberfläche des Werkzeugs übertragen und kollationiert, wobei die Laufzeit des Vergleichs durch die Modifikation automatisch mitbestimmt und ausgegeben wurde.

Anmerkungen zu LERA

Für den Test von LERA wurde eine zusätzliche Importfunktion ins System integriert, welche alle notwendigen Schritte zur Kollationierung eines Datensatzes abarbeitet. Dabei werden die Texte zunächst ins System geladen und tokenisiert, was gleichzeitig die grundlegenden Datenbankstrukturen für die Textfassungen anlegt. Anschließend werden die drei Schritte der Kollationierung (siehe Abschnitt 5.1) durchgeführt: eine Segmentierung anhand der in den Datensätzen als Zeilenumbrüche codierten Absätze, eine Alignierung dieser Absätze anhand ihrer Ähnlichkeit mittels des Chain-Graph-Signature-Aligners und schließlich ein Detailvergleich auf Tokenebene innerhalb der alignierten Absätze. Dabei wurden keine zusätzlichen Filter angewandt, aber alle notwendigen Daten berechnet, um die Textvarianten

[3] Unter: `https://github.com/ecomp-shONgit/ecomparatio` (besucht am 19.07.2022).

[4] Stichwort: Cross-Origin Resource Sharing.

[5] So wurden im ursprünglichen Quellcode Variablen durch Verwendung des Schlüsselworts var statt let (globale statt lokale Initialisierung) teilweise doppelt angelegt, was aufgrund möglicher Seiteneffekte von modernen Browsern oft nicht (mehr) toleriert wird.

anschließend als Spaltensynopse mit grauen Hervorhebungen und dem Variantenapparat in LERA anzuzeigen. Die angegebene Laufzeit misst die Ausführung der gesamten Importfunktion.

Ergebnisse

TRAViz

Wie bereits in [75] angemerkt, ist TRAViz nicht für umfangreiche Textmengen ausgelegt. So wird die gesetzte Grenze für die Laufzeit von einer Stunde auch bereits für die Datensätze *2_4000* und *4_4000* überschritten. Primärer Grund dafür ist der intern verwendete Brute-Force-Ansatz zur Alignierung, welcher einen rapiden Anstieg der Laufzeiten verursacht. Folgende Tabelle zeigt die ermittelten Laufzeiten getrennt für zwei und vier Textfassungen.

	2_500	*2_1000*	*2_2000*	*2_4000*
Alignierung	2,12	41,52	630,20	13.826,94
Visualisierung	1,79	8,50	55,02	223,84
Gesamt	3,91	50,01	685,22	14.050,78

	4_500	*4_1000*	*4_2000*	*4_4000*
Alignierung	7,70	169,86	3.088,11	x
Visualisierung	5,93	36,66	380,22	x
Gesamt	13,62	206,52	3.468,34	x

Tabelle A.1: Die Laufzeiten in Sekunden (auf zwei Nachkommastellen gerundet) für die Kollationierung unterschiedlich umfangreicher Datensätze mittels TRAViz. Die Berechnung für 4_4000 wurde nach mehreren Stunden abgebrochen.

CollateX

Die Experimente für die drei in CollateX integrierten Algorithmen – Dekker, Needleman-Wunsch (N-W) und MEDITE – führten zu sehr unterschiedlichen Resultaten. Mit dem Dekker-Algorithmus konnten Textmengen von 4.000 beziehungsweise 2.000 Wörtern noch im Sekundenbereich kollationiert werden, bevor die Laufzeiten

stark anstiegen und mit den Datensätzen 2_32000 und 4_16000 schließlich die gesetzte Grenze von einer Stunde überschritten. Der integrierte Needleman-Wunsch-Algorithmus (N-W) war zunächst effizienter für die mittelgroßen Datensätze, stürzte dann allerdings bei 2_32000 und 4_16000 aufgrund eines unbehandelten Fehlers (fehlender Speicherplatz im Heap) ab. Auch beim MEDITE-Algorithmus trat dieser Fehler bereits für den Datensatz 2_4000 auf, während die Berechnung für den Datensatz 4_2000 nach sechs Stunden (mögliche Endlosschleife) abgebrochen wurde. Tabelle A.2 listet die Ergebnisse im Einzelnen.

	2_500	*2_1000*	*2_2000*	*2_4000*	*2_8000*	*2_16000*	*2_32000*
Dekker	0,95	1,26	2,27	6,49	77,05	2.156,81	28.685,18
N-W	1,11	1,35	2,04	4,88	16,63	69,34	x
MEDITE	1,55	3,31	6,62	30,13	x	-	-

	4_500	*4_1000*	*4_2000*	*4_4000*	*4_8000*	*4_16000*	*4_32000*
Dekker	1,22	2,51	10,90	108,57	1.189,87	22.143,00	-
N-W	1,41	2,01	4,90	16,91	251,19	x	-
MEDITE	2,31	4,92	x	-	-	-	-

Tabelle A.2: Die Laufzeiten in Sekunden für die Kollationierung unterschiedlich umfangreicher Datensätze mit je zwei und vier Textfassungen für die drei in CollateX integrierten Algorithmen. Die Werte wurden auf zwei Nachkommastellen gerundet. Der Eintrag „x“ zeigt das Auftreten eines Programmfehlers an, der Eintrag „-“ ein nicht ausgeführtes Experiment aufgrund des vorherigen Fehlschlags mit kleineren Datensätzen.

eComparatio

Nach dem Beheben einiger Probleme (siehe oben) konnten mit eComparatio Ergebnisse für alle Datensätze bestimmt werden. Während die Berechnung für kleine bis mittlere Textmengen bei zwei Textfassungen dabei zunächst sehr schnell erfolgte, stiegen die Laufzeiten mit zunehmender Textmenge stark an und erreichten schließlich über 4 Minuten für 2_32000 und rund 7 Minuten für 2_all. Gleiches gilt für die Experimente mit vier Textfassungen, welche in Laufzeiten um die 20 Minuten mündeten. Die Ergebnisse werden im Folgenden aufgeschlüsselt.

2_500	*2_1000*	*2_2000*	*2_4000*	*2_8000*	*2_16000*	*2_32000*	*2_all*
0,16	0,42	1,14	3,73	14,36	63,57	255,78	415,16

4_500	*4_1000*	*4_2000*	*4_4000*	*4_8000*	*4_16000*	*4_32000*	*4_all*
0,94	2,40	7,88	26,16	94,34	375,26	1.181,31	1.331,49

Tabelle A.3: Die Laufzeiten in Sekunden für die Kollationierung unterschiedlich umfangreicher Datensätze mit je zwei oder vier Textfassungen mittels eComparatio. Die Werte wurden auf zwei Nachkommastellen gerundet.

LERA

Mit LERA konnten alle Datensätze unterhalb der gesetzten Grenze von einer Stunde verarbeitet und kollationiert werden. Tatsächlich lag die Laufzeit für den größten Datensatz 4_all, welcher die vier Textfassungen des sechsten Buches der *Histoire des deux Indes* in ihrer Gesamtlänge enthält, bei unter vier Minuten. Die folgende Tabelle listet die Laufzeiten im Detail.

	2_500	*2_1000*	*2_2000*	*2_4000*	*2_8000*	*2_16000*	*2_32000*	*2_all*
Segmentierung	0,30	0,49	0,61	0,95	1,75	3,10	5,91	9,01
Alignierung	0,20	0,20	0,35	0,62	1,21	3,22	7,52	12,86
Detailvergleich	0,20	0,25	0,59	0,99	1,68	3,72	7,02	10,72
Gesamt	1,93	2,33	4,06	7,11	14,41	32,35	78,93	130,38

	4_500	*4_1000*	*4_2000*	*4_4000*	*4_8000*	*4_16000*	*4_32000*	*4_all*
Segmentierung	0,53	0,59	1,04	1,81	3,35	6,10	11,13	13,98
Alignierung	0,30	0,40	0,73	1,44	3,65	10,96	26,39	37,03
Detailvergleich	0,34	0,72	1,07	2,95	4,68	10,87	18,72	24,33
Gesamt	3,50	4,44	7,64	15,34	30,56	73,23	159,37	219,01

Tabelle A.4: Die Laufzeiten in Sekunden für die Kollationierung unterschiedlich umfangreicher Datensätze mit je zwei und vier Textfassungen in LERA. Die Werte wurden auf zwei Nachkommastellen gerundet. Neben der Gesamtlaufzeit werden auch die darin enthaltenen Laufzeiten der drei Schritte zur hierarchischen Kollationierung (Segmentierung, Alignierung und Detailvergleich) aufgeführt.

In Tabelle A.4 werden neben der Gesamtlaufzeit auch die darin enthaltenen Laufzeiten der drei Schritte zur hierarchischen Kollationierung (Segmentierung, Alignierung und Detailvergleich) einzeln aufgeführt. Für alle Datensätze machten diese in Summe weniger als die Hälfte der Gesamtlaufzeit aus. Statt der eigentlichen Kollationierung verursachte folglich die initiale Verarbeitung der Textfassungen in Form der Tokenisierung sowie dem Anlegen der grundlegenden Datenbankstrukturen den Großteil der in den Experimenten bestimmten Laufzeiten für LERA.

Zusammenfassung und Anmerkungen

Folgende Tabelle zeigt die ermittelten Gesamtlaufzeiten für die untersuchten Werkzeuge als Übersichtstabelle, wobei die drei in CollateX integrierten Algorithmen (Dekker, Needleman-Wunsch und MEDITE) einzeln aufgeführt werden.

	2_500	*2_1000*	*2_2000*	*2_4000*	*2_8000*	*2_16000*	*2_32000*	*2_all*
TRAViz	4	50	685	14.051	-	-	-	-
Dekker	1	1	2	6	77	2.157	28.685	-
N-W	1	1	2	5	17	69	x	-
MEDITE	2	3	7	30	x	-	-	-
eComparatio	0	0	1	4	14	63	256	415
LERA	2	2	4	7	14	32	79	130

	4_500	*4_1000*	*4_2000*	*4_4000*	*4_8000*	*4_16000*	*4_32000*	*4_all*
TRAViz	14	207	3.468	x	-	-	-	-
Dekker	1	3	11	109	1.190	22.143	-	-
N-W	1	2	5	17	251	x	-	-
MEDITE	2	5	x	-	-	-	-	-
eComparatio	1	2	8	26	94	375	1.181	1.331
LERA	4	4	8	15	31	73	159	219

Tabelle A.5: Übersichtstabelle der ermittelten Gesamtlaufzeiten verschiedener Ansätze zur Kollationierung. Die Laufzeiten wurden dabei auf volle Sekunden gerundet. Der Eintrag „x“ zeigt das Auftreten eines Programmfehlers oder einen Abbruch an, der Eintrag „-“ ein nicht ausgeführtes Experiment aufgrund des vorherigen Fehlschlags mit kleineren Datensätzen.

Während TRAViz nur für kleine Textmengen ausgelegt ist, was sich in den Experimenten bestätigte, ermöglichte CollateX die Kollationierung der mittelgroßen Datensätze. Allerdings ist bei CollateX die Wahl des Algorithmus zu beachten: Am robustesten hat sich der Dekker-Algorithmus gezeigt, während die integrierten Implementierungen des Needleman-Wunsch- sowie MEDITE-Algorithmus teilweise schneller waren, ab einem gewissen Textumfang allerdings zu Programmfehlern führten. Mit eComparatio konnten alle Datensätze des Experiments innerhalb der gesetzten Grenze von einer Stunde kollationiert werden. Die Laufzeiten für große Textmengen stiegen dabei auf über 20 Minuten an. Auch mit LERA konnten alle Datensätze kollationiert werden. Die Laufzeiten blieben dabei unterhalb von vier Minuten.

Bei dem in diesem Anhang beschriebenen Experimenten und ihrer Auswertung handelt es sich um keine vergleichende Studie der verschiedenen Werkzeuge. Zudem spielte die Qualität der berechneten Kollationierungsergebnisse keine Rolle, zumal die Werkzeuge unterschiedliche Ansätze verfolgen und damit nur bedingt vergleichbare Ergebnisse liefern. Der Fokus lag stattdessen auf der Frage, inwieweit die Kollationierung umfangreicher Textfassungen mit den Werkzeugen möglich ist. Aus diesem Grund wurde auch je nur eine Messung pro Datensatz durchgeführt, denn mögliche Schwankungen im Sekundenbereich sind für die Beantwortung der untersuchten Frage nicht von Relevanz.

Erklärung

Für die Umsetzung eines so umfangreichen Kollationierungswerkzeugs wie LERA, welches für den praktischen Einsatz in verschiedenartigen Editionsprojekten unterschiedlicher philologischer Fachrichtungen gedacht ist, war und ist eine stetige geisteswissenschaftliche Perspektive auf Problemlösungen und damit eine enge Kooperation mit Kolleg:innen aus diesen Gebieten unabdingbar. Ohne diese interdisziplinäre Zusammenarbeit wäre LERA nicht das, was es jetzt ist: Eine auf internationalen Fachtagungen diskutierte Software, welche in dutzenden Editionsvorhaben eingesetzt wird (siehe Abschnitt 7.1.2). So haben der Dialog mit Kolleg:innen aus verschiedenen Disziplinen und ihr hilfreiches Feedback die Realisierung von LERA in der jetzigen Form erst ermöglicht. Zu nennen sind hierbei insbesondere Susanne Schütz und Prof. Dr. Thomas Bremer vom Institut für Romanistik der Martin-Luther-Universität Halle-Wittenberg, Prof. Dr. Dr. h.c. Beatrice Gründler und Mahmoud Kozae des Seminars für Semitistik und Arabistik der Freien Universität Berlin, Dr. Manuela Bragagnolo und Dr. Andreas Wagner vom Max-Planck-Institut für Rechtsgeschichte und Rechtstheorie in Frankfurt am Main, Dr. Brigitte Grote und Fabian Etling des Centers für Digitale Systeme (CeDiS) der Freien Universität Berlin, sowie Dr. Bill Rebiger und apl. Prof. Dr. Gerold Necker vom Seminar für Judaistik/Jüdische Studien der Martin-Luther-Universität Halle-Wittenberg. Gleiches gilt für meine Betreuer Dr. Jörg Ritter und Prof. Dr. Paul Molitor vom Institut für Informatik der Martin-Luther-Universität Halle-Wittenberg.

Wie in Kapitel 4 ausgeführt, geht LERAs Ursprung auf das interdisziplinäre Forschungsprojekt *Semi-automatische Differenzanalyse von komplexen Textvarianten (SaDA)* zurück, welches aus zwei eng miteinander verknüpften Teilprojekten bestand. Zum einen wurde die Kollationierung umfangreicher Druckwerke am Beispiel von vier Textfassungen der *Histoire des deux Indes* erforscht, aus der im Laufe des Projekts LERA hervorging. Zum anderen wurde im zweiten Teilprojekt die Kollationierung eines frühneuhochdeutschen, handschriftlichen Manuskripts in zehn Fassungen (Die *Wundarznei* des Heinrich von Pfalzpaint) untersucht und im Rahmen dessen die Software LAKomp entwickelt. Beide Teilprojekte haben dabei insbesondere in der Informatik zusammengearbeitet und die erarbeiteten Lösungen ausgetauscht. Auf diesem Weg wurde der von André Medek (né Gießler) für LAKomp konzipierte und implementierte Chain-Graph-Signature-Aligner (siehe Abschnitt 5.3.2) mit kleineren Modifikationen und Erweiterungen in LERA übernom-

men, der dabei zur Alignierung von Textsegmenten genutzt wird. Um ein vollständiges Bild der algorithmischen Arbeitsweise von LERA in dieser Dissertation zu zeichnen, wird das grundlegende Vorgehen des Chain-Graph-Signature-Aligners unter Nennung des eigentlichen Autors in Abschnitt 5.3.2 vorgestellt. Umgekehrt wurde wiederum die ursprünglich für LERA entwickelte CATview (siehe Abschnitt 6.2) in LAKomp integriert.

Die in LERA integrierten Wortwolken wurden 2019 um eine verbesserte Realisierung stabiler Wortpositionen erweitert (siehe Abschnitt 6.3), welche den vorherigen Prototypen mit der nun in LERA verwendeten und in dieser Dissertation vorgestellten Form ersetzt hat. Die algorithmische Grundlage geht dabei auf eine studentische Arbeit von Elisa Herold zurück, die zusammen mit weiteren Kollegen sukzessive verbessert und publiziert wurde (siehe [63] und [32]). Mein Beitrag zur jetzigen Form der stabilen Wortwolken besteht in der Einbindung in LERA sowie der Erweiterung auf eine fassungsübergreifende Datengrundlage, um die in LERA synoptisch angeordneten Wortwolken verschiedener Textfassungen miteinander zu verknüpfen und vergleichbar zu gestalten.

Als Grundlage für die Struktur und Formulierung einiger Abschnitte dieser Dissertation habe ich vier Publikationen von den insgesamt acht eigenen Zeitschrifteinartikeln beziehungsweise Buchbeiträgen (davon vier als Erstautor) sowie den 18 Konferenzbeiträgen genutzt. So ist der in Abschnitt 4.2 ausgeführte Abgleich von LERAs Arbeitsweise und Funktionalitäten mit dem Gothenburg-Modell eine erweiterte Fassung von [118, S.334-340]. Teile von [118] sind zudem in weitere Abschnitte der Dissertation eingeflossen, insbesondere in Abschnitt 3.2. Die Beschreibung von CATview in Abschnitt 6.2 basiert auf einem Konferenzartikel von 2015 ([117]). Ebenso wurde ein Konferenzartikel von 2016 ([139]) als Grundlage für die Ausführungen in Kapitel 6 zum Zusammenspiel der interaktiven Visualisierungen in LERA genutzt. Hieraus stammt auch das in Abbildung 6.8 illustrierte Beispiel. In Abschnitt 6.4 zur Explorativen Analyse innerhalb des Gesamtkorpus werden verschiedene, in LERA integrierte Visualisierungen vorgestellt. Hierbei habe ich Teile meiner 2024 erscheinenden Publikation [115] als Grundlage der Ausführungen genutzt.

Eidesstattliche Erklärung / *Declaration under Oath*

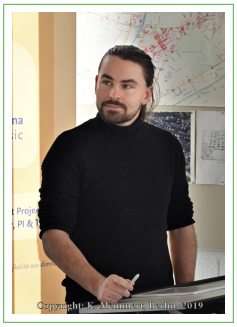
Ich erkläre an Eides statt, dass ich die Arbeit selbstständig und ohne fremde Hilfe verfasst, keine anderen als die von mir angegebenen Quellen und Hilfsmittel benutzt und die den benutzten Werken wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

I declare under penalty of perjury that this thesis is my own work entirely and has been written without any help from other people. I used only the sources mentioned and included all the citations correctly both in word or content.

25.07.2024

Datum / *Date*

Unterschrift des Antragstellers / *Signature of the applicant*



Marcus Pöckelmann

Akademischer Lebenslauf

Bildungsweg

2003–2006 Allgemeine Hochschulreife
Gymnasium BbSII Guttjahr, Halle (Saale)

2007–2010 Bachelor of Science (Informatik)
Martin-Luther-Universität Halle-Wittenberg (MLU)
Bachelorarbeit zum Thema „*Ist die Welt kleiner geworden?* Experimentelle Analyse des tageszeitabhängigen Durchmessers im Flugverkehr“, betreut durch Prof. Dr. Matthias Müller-Hannemann und Annabell Berger

2010–2013 Master of Science (Informatik)
Martin-Luther-Universität Halle-Wittenberg (MLU)
Masterarbeit zum Thema „Zyklizität in empirischen und künstlichen Nahrungsnetzen“, betreut durch Prof. Dr. Matthias Müller-Hannemann und Dr. Annabell Berger

Wissenschaftlicher Mitarbeiter an der MLU

2012-2015 Forschungsprojekt: *SaDA - Semi-automatische Differenzanalyse von komplexen Textvarianten* (Drittmittel, BMBF)

WS2015/16 Haushaltsstelle zur Unterstützung der Lehr- und Forschungsaufgaben in dem Gebiet der Informatik in den Geistes- und Sozialwissenschaften

2016-2019 Forschungsprojekt: *Platon Digital: Tradition and Reception* (Drittmittel, VolkswagenStiftung)

2019-2025 Forschungsprojekt: *Synoptische Edition des kabbalistischen Traktats Keter Shem Tov mit englischer Übersetzung, Stellenkommentar und rezeptionsgeschichtlichen Studien: Aufbau einer digitalen Mehrschicht-Synopse und alphabet-mystische Traditionen zum kabbalistischen Traktat Keter Shem Ṭov.* (Drittmittel, DFG)

Buch- und Zeitschriftenartikel

2015 A. Medek, M. Pöckelmann, T. Bremer, H.-J. Solms, P. Molitor und J. Ritter: „Differenzanalyse komplexer Textvarianten - Diskussion und Werkzeuge“. In: G. Heyer und A. Henrich (eds.) *Datenbank-Spektrum - Informationsmanagement für Digital Humanities*, 2015;15(1):25-31. https://doi.org/10.1007/s13222-014-0173-y

2015 T. Bremer, P. Molitor, M. Pöckelmann, J. Ritter und S. Schütz: „Zum Einsatz digitaler Methoden bei der Erstellung und Nutzung genetischer Editionen gedruckter Texte mit verschiedenen Fassungen - Das Fallbeispiel der Histoire philosphique des deux Indes von Guillaume Thomas Raynal“. In: R. v. Nutt-Kofoth, B. Plachta und W. Woesler (eds.) *Editio*, 2015;29(1):29-51. https://doi.org/10.1515/editio-2015-004

2017 M. Pöckelmann, J. Ritter, E. Wöckener-Gade und Ch. Schubert: „Paraphrasensuche mittels word2vec und der Word Mover's Distance im Altgriechischen“. In: *Digital Classics Online*, 2017;3(3):24-36. https://doi.org/10.11588/dco.2017.0.40185

2019 M. Pöckelmann, J. Ritter und P. Molitor: „Word Mover's Distance angewendet auf die Paraphrasenextraktion im Altgriechischen“. In: C. Schubert, P. Molitor, J. Ritter, J. Scharloth und K. Sier (eds.) *Platon Digital: Tradition und Rezeption*. Heidelberg: Propylaeum, (Digital Classics Books, 2019:3). https://doi.org/10.11588/propylaeum.451

2020 M. Pöckelmann, J. Dähne, J. Ritter und P. Molitor: „Fast paraphrase extraction in Ancient Greek literature“. In: *it - Information Technology*, 2020;62(2):75–89. https://doi.org/10.1515/itit-2019-0042

2021 J. Dähne, M. Pöckelmann, J. Ritter und P. Molitor: „Putting collation of text witnesses on a formal basis“. In: *Digital Scholarship in the Humanities*, 2022;37(2):375–390. https://doi.org/10.1093/llc/fqab058

2022 M. Pöckelmann, A. Medek, J. Ritter und P. Molitor: „LERA - An interactive platform for synoptical representations of multiple text witnesses“. In: *Digital Scholarship in the Humanities*, 2023;38(1):330–346. https://doi.org/10.1093/llc/fqac021

2023 E. Wöckener-Gade und M. Pöckelmann: „Innovation in Loops: Developing Tools and Redefining Theories within the Project 'Digital Plato'“. In: B. Schneider, B. Löffler, T. Mager und C. Hein (eds.) *Mixing Methods: Practical Insights from the Humanities in the Digital Age*. Bielefeld: transcript Verlag, Digital Humanities Research, Vol. 7:47-60. https://doi.org/10.14361/9783839469132-006

Konferenzbeiträge

2014 A. Gießler, M. Pöckelmann und J. Ritter: „Semi-automatische Differenzanalyse von komplexen Textvarianten“. *DHd2014, 1. Jahrestagung der Digital Humanities im deutschsprachigen Raum*, Passau, 25.-28.03.2014. https://doi.org/10.5281/zenodo.4623507

2014 M. Pöckelmann, J. Ritter und A. Medek: „On automatically disambiguating end-of-line hyphenated words in French texts“. *DH2014, 25. international annual conference of Digital Humanities*, Lausanne, 07.-12.07.2014.

2014 S. Schütz und M. Pöckelmann: „IT-Werkzeuge zur Unterstützung elektronischer Edition am Beispiel eines französischen Textes aus dem 18. Jahrhundert“. „9. Kongress des Frankoromanistenverbands“, Münster, 24.-27.09.2014.

2015 A. Medek, M. Pöckelmann et al.: „SaDA – Werkzeuge für die semi-automatische Differenzanalyse komplexer Textvarianten“. *DH Summit 2015*, Berlin, 03.-04.03.2015.

2015 M. Pöckelmann, A. Medek, P. Molitor und J. Ritter: „CATview - Supporting The Investigation Of Text Genesis Of Large Manuscripts By An Overall Interactive Visualization Tool“. *DH2015, 26. international annual conference of Digital Humanities*, Sydney, 29.06.-03.07.2015.

2016 S. Schütz und M. Pöckelmann: „LERA - Explorative Analyse komplexer Textvarianten in Editionsphilologie und Diskursanalyse“. *DHd2016, 3. Jahrestagung der Digital Humanities im deutschsprachigen Raum*, Leipzig, 07.-12.03.2016. https://doi.org/10.5281/zenodo.4645364

2017 R. Kath, F. Keilholz, F. Klinker, M. Pöckelmann, M. Rücker, M. Švitek, E. Wöckener-Gade und X. Yu: „Paraphrasenerkennung im Projekt Digital Plato“. *DHd2017, 4. Jahrestagung der Digital Humanities im deutschsprachigen Raum*, Bern, 13.-18.02.2017. https://doi.org/10.5281/zenodo.4622702

2018 R. Kath, F. Keilholz, M. Pöckelmann, M. Rücker, E. Wöckener-Gade und X. Yu: „Entwicklungsstand im Projekt Digital Plato“. *DHd2018, 5. Jahrestagung der Digital Humanities im deutschsprachigen Raum*, Köln, 26.02-02.03.2018. https://doi.org/10.5281/zenodo.4622522

2018 B. Gründler und M. Pöckelmann: „Adjusting LERA For The Comparison Of Arabic Manuscripts Of Kalīla wa-Dimna“. *DH2018, 29. international annual conference of Digital Humanities*, Mexico City, 26.-29.06.2018.

2018 E. Wöckener-Gade und M. Pöckelmann: „Bridging the Gap between Plato and His Successors: Towards an Annotated Gold Standard of Intertextual References to Plato in Ancient Greek Literature“. *EADH 2018, Data in Digital Humanities*, Galway, 07.-09.12.2018.

2019 E. Herold, M. Pöckelmann, C. Berg, R. Ritter und M. M. Hall: „Stable Word-Clouds for Visualising Text-Changes Over Time“. In: A. Doucet, A. Isaac, K. Golub, T. Aalberg und A. Jatowt (eds.) *Digital Libraries for Open Knowledge - Proceedings of the 23. International Conference on Theory and Practice of Digital Libraries, TPDL2019*, Oslo, 09.-12.09.2019. pp. 224-237. https://doi.org/10.1007/978-3-030-30760-8_20

2020 P. Molitor, G. Necker, M. Pöckelmann, B. Rebiger und J. Ritter: „Keter Shem Ṭov Prozessualisierung eines Editionsprojekts mit 100 Textzeugen“. *DHd2020, 7. Jahrestagung der Digital Humanities im deutschsprachigen Raum*, Paderborn, 02.-06.03.2020. https://doi.org/10.5281/zenodo.4621883

2022 M. Pöckelmann und B. Rebiger: „Anpassungen von LERA zum Vergleich hebräischer Textzeugen des kabbalistischen Traktats Keter Shem Ṭov“. *DHd2022, 8. Jahrestagung der Digital Humanities im deutschsprachigen Raum*, Potsdam, 07.-11.03.2022. https://doi.org/10.5281/zenodo.6328132

2022 T. K. H. Luu, M. Pöckelmann, J. Ritter und P. Molitor: „Applying LERA for collating witnesses of The Tale of Kiều, a Vietnamese poem written in Nôm script“. *DH2022, 32. international annual conference of Digital Humanities*, Tokyo, 25-29.07.2022.

2022 B. Rebiger und M. Pöckelmann: „The Edition of Keter Shem Ṭov by means of the tool LERA (Locate, Explore, Retrace and Apprehend complex text variants“. *The 18th World Congress of Jewish Studies*, Jerusalem, 08.-12.08.2022. https://program.eventact.com/Lecture/243250/5487004

2023 J. Dähne, M. Pöckelmann und J. Ritter: „Word Clouds with Spatial Stable Word Positions across Multiple Text Witnesses“. *DH2023, 33. international annual conference of Digital Humanities*, Graz, 10-14.07.2023.

2024 F. Etling, M. Pöckelmann und B. Grote: „Auffinden und Analysieren komplexer Textvarianten in Hannah Arendts Denk- und Schreibwerkstatt mit LERA“. *DHd2024, 10. Jahrestagung der Digital Humanities im deutschsprachigen Raum*, Passau, 26.02.-01.03.2024.

2024 M. Pöckelmann: „Extensions of the Digital Collation Tool LERA for the Scholarly Edition of Keter Shem Ṭov“. In: G. Necker und B. Rebiger (eds.) *Proceedings of the Workshop Editing Kabbalistic Texts*. Harrassowitz, erscheint 2024.