Softwaretools für die Literaturtextanalyse – Ein Überblick

Hermann Johannes (Informatik)

1. Einleitung

Der Artikel behandelt aus dem Bereich Digital Humanities das Thema Literatur und Informatik und untersucht als Schwerpunkt, welche Anforderungen zur Analyse von Literaturtexten (bisher) vorhanden sind, welche Methoden zur Lösung existieren und welche Softwaretools zur Erfüllung dieser Anforderungen genutzt werden. Diese Tools werden kurz beschrieben – vom Download über die Installation bis hin zu den Funktionen der Anwendung – und auch kurz bewertet. Neben den Tools werden einige Projekte und Studien beschrieben, die sich in jüngerer Vergangenheit mit Literaturtextanalyse auseinandergesetzt und oft mehrere der o.g. Tools eingesetzt haben. Ergänzend werden die Themen methodisches Vorgehen zur Erstellung eines Softwareprogramms, IT-Begriffe, Textcodierung, Methoden der Literaturtextanalyse und statistische Merkmale eines literarischen Textes behandelt.

Der Artikel erhebt keinen Anspruch auf Vollständigkeit, da die Literaturtextsoftware inzwischen ein nahezu unüberschaubares Ausmaß erreicht hat. Als Zielgruppe ist er eher für Einsteiger:innen in die DH gedacht, die z.B. aus einer der Disziplinen der Geisteswissenschaften kommen und sich in Richtung DH weiter informieren möchten. Für diese Aufgabenstellung soll der Artikel eine Art Dokumentation sein und darüber hinaus auch Anreize geben, sich mit den Methoden, Tools und Projekten aus diesem Feld auseinander zu setzen und die Tools auch selbst zu nutzen.

Die Informatik ist eigentlich eine Hilfswissenschaft (wie die Mathematik), die keinen Selbstzweck hat, sondern Services für die diversen Anwendungsbereiche wie z.B. Buchhaltung, Produktionsplanung, Medizindiagnose oder Analyse von literarischen Texten zur Nutzung durch den/die Anwender:in bietet. Die viel zitierte digitale Transformation durchdringt immer stärker alle Bereiche (Arbeitswelt, Kultur, Bildung, Freizeit, Gesellschaft), somit auch die der Literatur und der Kunst. Unsere Gesellschaft ist davon in einem ähnlichen Ausmaß betroffen, wie durch die industrielle Revolution in der zweiten Hälfte des 18. Jahrhunderts.

Die Informatik ist eine Wissenschaft, bei der Daten/Informationen ermittelt, strukturiert, gespeichert, verarbeitet, übertragen und dargestellt werden. Um diese Prozesse zu bewerkstelligen, werden i.d.R. Computer (Personal Computer, Server, Tablet, Handy u.a.) in jeglicher Form eingesetzt. Sie ist einerseits eine Grundlagen- und Formalwissenschaft und gilt andererseits auch als Ingenieurdisziplin. Da die Informatik zur Speicherung der Daten digitale Codes auf Basis des Binärsystems (Bits und Bytes) nutzt, hat sich hier der Begriff Digitalisierung durchgesetzt.

Ein Beispiel für Softwareprogramme sind Buchhaltungssysteme. Diese reichen von einem einfachen Buchhaltungssystem bis hin zu komplexen Enterprise Ressource Planning Systemen (ERP), die fast alle Bereiche eines Konzerns abdecken können, von der Auftragsbearbeitung über die Produktion und Logistik bis hin zum Vertrieb (Beispiel SAP©).

Weitere aktuelle Entwicklungen der Digitalisierung sind die Arbeitswelt 4.0, künstliche Intelligenz (KI), wie u.a. in Robotern für Produktion, Haushalt und Altenpflege oder autonomes Fahren, Bildgebungsverfahren in der Medizin, *Virtual* oder *Augmented Reality* (Beispiel: Nutzung einer Brille, die bei einer Reparatur die zu der Tätigkeit passende Anleitung einblendet), Desktop Publishing und Bildverarbeitung im Journalismus, Spracherkennungs- und Übersetzungssysteme, Gerätesteuerung im Heimbereich (Smart Home), Computerspiele u.v.a.m. In PCs, Notebooks, Tablets und Smartphones unterstützen immer mehr Apps bei der täglichen Arbeit oder Freizeitgestaltung. Ein KI-Tool wird aktuell besonders viel diskutiert: ChatGPT (Generative Pre-trained Transformer), ein sogenannter Chatbot, also ein textbasiertes Dialogsystem, das auf maschinellem Lernen basiert. Bei der KI-Forschung hat sich ein Wettbewerb der Nationen entwickelt, wobei viele der Protagonisten im Silicon Valley ihren Firmensitz haben, z.B. Google. Aber auch in Deutschland hat die KI-Forschung einen hohen Stellenwert, so z.B. bei der Firma DeepL in Köln, die das derzeit weltbeste Übersetzungsprogramm (www.deepl.com) entwickelt hat.

Die vielfältigen Kommunikationsmöglichkeiten über das Internet (inzwischen auch das semantic web) und mithilfe Sozialer Medien haben in den beiden letzten Jahrzehnten die Nutzung der Digitalisierung vorangetrieben. Dabei konnte die KI-Nutzung sich auch deshalb gut entwickeln, weil die immer weiter verbesserte Hardware (Prozessorleistung, Computernetzwerke und Übertragungsraten, größere und schnellere Speicher wie SSD-Platten) es nun möglich macht, riesige Datenmengen (Big Data) performant (mit hoher Geschwindigkeit) und damit zeitunkritisch zu verarbeiten.

Die Informatik nutzt in ihren unterstützenden Softwareprogrammen komplexe Algorithmen, die teilweise sogar natürliche Prozesse nachbilden. Dazu gehören u.a. Neuronale Netze, Fuzzytechnologien, Deep Learning und Evolutionäre Algorithmen (survival of the fittest). Inzwischen sind Computerprogramme so leistungsfähig, dass sie einem Profischachspieler oder einem Go-Spieler überlegen sind.

Durch die Digitalisierung fallen einige Jobs und Berufsfelder weg, andere entstehen. In einer Reihe von Studien wird diese Situation inhaltlich und statistisch beschrieben (vgl. Dämon 2022, S. 2f). Bereits 2015 gingen Prognosen u.a. der Süddeutschen Zeitung von ca. »50 % der Jobs in 700 Berufsgruppen aus, die in den nächsten 20 Jahren bedroht sind.« (Weyrauch 2015, S. 1). Betroffen sind auch viele Einzelhandelsunternehmen, die wegen des immer weiter zunehmenden Online-Handels wegfallen, wenn sie nicht er-

folgreich mit den großen Unternehmen wie Amazon© oder Otto© im Wettbewerb mithalten können.[1]

2. Begriffe, Methoden und Vorgehensweisen (IT)

2.1 Methodisches Vorgehen zur Erstellung eines Softwareprogramms

Mit der Entwicklung eines neuen Softwareprogramms – oder der Weiterentwicklung eines vorhandenen – beschäftigt sich in der Informatik die Disziplin *Software Engineering*. Dabei hat sich seit vielen Jahren ein phasenweises Vorgehen als zielführend herausgestellt, wobei folgende Phasen durchlaufen werden:

- Anforderungsanalyse (Definition von Anforderungen)
- Istanalyse mit Erfassung von Schwachstellen, falls schon eine Lösung (Softwareprogramm) existiert
- Sollkonzept (Pflichtenheft) und Systementwurf
- Umsetzung (Datenbank anlegen, Programmierung, erste Testdaten erstellen)
- Test (die Tests sollten weitgehend automatisiert wiederholt werden können, wenn Änderungen im Programm gemacht werden)
- Einführung, Dokumentation, Support und Wartung

Bei den ersten drei Phasen wird zwischen Fach- und DV-Konzept (IT-Konzept) unterschieden. Das Fachkonzept ist die Sicht der Anwendung: welche Eigenschaften (Funktionen und Benutzeroberfläche) hat das Programm. Das DV-Konzept konkretisiert die softwaretechnische Umsetzung: Plattform (Betriebssystem, Datenbank, Programmiersprache), Schnittstellen, Struktur der Anwendung (Datenmodell, Funktionsmodell) usw. Um das Softwareprogramm zunächst nicht zu umfangreich zu gestalten, was oft zur Einstellung des Projekts aus Kosten- oder Zeitgründen geführt hat, wird erst ein kleiner Teil der gewünschten Funktionen in Form eines Prototyps umgesetzt. Nach dem Test dieses Prototyps durch die Nutzer:innen und Korrekturen wird dann die nächste Stufe der Anwendung mit weiteren Funktionen entwickelt und so fort. Als Entwicklungsmodell kann das Wasserfallmodell (vgl. Royce 1970) mit iterativer Erweiterung (im Anschluss an Boehm 1986) genutzt werden, d.h. das eigentlich lineare Wasserfallmodell wird dahingehend erweitert, dass es mehrmals (iterativ) durchlaufen wird, um die Anforderungen sukzessive zu erweitern oder um aus Tests resultierende Korrekturen vorzunehmen. Wichtige Punkte für eine erfolgreiche Software sind auch eine gute und anwendergerechte Dokumentation, eine Hotline für Hilfestellungen sowie Support und Wartung des Programms.

1 Zusätzliche Texte und praktische Analysen zu diesem Band werden auf einem Downloadserver bei GitHub zur Verfügung gestellt: https://github.com/LuckeJohannes/Literaturwissenschaft_Informatik. Dazu gehören Artikel zur Stilometrie, zur KI-Clusteranalyse von Autor:innen sowie ein Glossar und ein Artikel zu Spezialthemen.

Neben dem (erweiterten) Wasserfallmodell haben sich weitere Software-Engineering-Modelle bewährt u.a. das iterative Spiralmodell[2] oder die agile Softwareentwicklung.[3] Letztere besteht aus sog. vier Werten und zwölf Prinzipien. Die vier Werte sind:

- »Individuen und Interaktionen mehr als Prozesse und Werkzeuge
- Funktionierende Software mehr als umfassende Dokumentation
- Zusammenarbeit mit dem Kunden mehr als Vertragsverhandlung
- Reagieren auf Veränderung mehr als das Befolgen eines Plans« (Ebd.)

Die 12 Prinzipien sind u.a., »den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software zufrieden zu stellen« und »heiße Anforderungsänderungen selbst spät in der Entwicklung willkommen«[4] zu heißen. »Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden.«[5]

In der Anforderungsanalyse wird beschrieben, welche fachlichen Anforderungen an ein Softwareprogramm gestellt werden, um daraus die Funktionen des Programms abzuleiten. Eine der Anforderungen ist die Mehrsprachigkeit der Benutzeroberfläche. Die Anwender:innen (Literaturwissenschaftler:innen) stellen Anforderungen, Programmierer:innen erstellen dazu passend die Softwaretools. Die Anforderungen sollten klar und eindeutig sein; um das zu gewährleisten, sollte eine toolgestützte Anforderungsanalyse durchgeführt werden.

Bei der Istanalyse wird untersucht, welche Funktionen schon in evtl. vorhandenen Programmen existieren, wobei auch vorhandene Schwachstellen ermittelt werden, die oft in der Benutzeroberfläche zu finden sind.

Das Sollkonzept beschreibt den kompletten Funktionsumfang des Programms, wobei ein Stufenplan die Schritte der Realisierung vorgibt. Es wird auch als Pflichtenheft bezeichnet.

In der Umsetzung wird das Softwareprogramm schließlich programmiert.

> Ob die Umsetzung einem Wasserfallmodell oder der modernen agilen Methode folgt, ist von Projekt zu Projekt unterschiedlich. Agile Methoden mit iterativem Vorgehen setzen sich in der Softwareentwicklung in den letzten Jahren immer stärker durch.[6]

Anschließende Tests stellen die fehlerfreie Nutzung des Softwareprogramms sicher. Schließlich wird eine (mehrsprachige) Dokumentation erstellt und Support und Wartung für die zukünftige Weiterentwicklung eingerichtet.

---

2 URL: https://www.computer.org/portal/cms_docs_computer/computer/homepage/misc/Boehm/r5061.pdf, [Zugriff: 20.11.2023]

3 Vgl. URL: https://wirtschaftslexikon.gabler.de/definition/agile-softwareentwicklung-53460/version-276549, [Zugriff: 20.11.2023]

4 URL: https://wirtschaftslexikon.gabler.de/definition/agile-softwareentwicklung-53460/version-276549, [Zugriff: 20.11.2023]

5 Vgl. URL: https://wirtschaftslexikon.gabler.de/definition/agile-softwareentwicklung-53460/version-276549, [Zugriff: 20.11.2023]

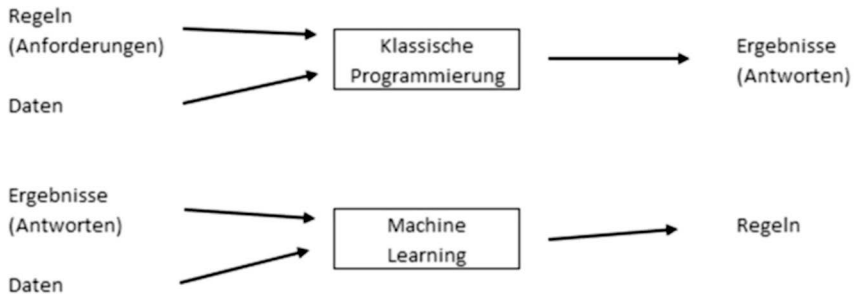
6 URL: https://bmu-verlag.de/spieleentwicklung/ [Zugriff: 05.11.2023], vgl. dazu auch Haufe 2022.

Häufig werden grafische Visualisierungen auf der Basis von geeigneten Modellierungssprachen wie UML (Unified Modelling Language) genutzt, die die Ergebnisse der einzelnen Phasen für die Anwender:innen bzw. Entwickler:innen transparenter machen.

Manche Anwendungen werden nicht mehr im klassischen Stil wie oben skizziert »programmiert«. »Bei der klassischen Programmierung […] geben Menschen Regeln (ein Programm) und die gemäß diesen Regeln zu verarbeitenden Daten vor, was zu Antworten führt«[7], im Sinne von Ergebnissen.

> Beim Machine Learning, einem Teilgebiet der Künstlichen Intelligenz (KI), geben Menschen sowohl die Daten als auch die dazugehörigen Antworten vor, und heraus kommen die Regeln. Diese Regeln sind dann auf neue Daten anwendbar und liefern eigenständige Antworten.[8]

Abb. 1: Machine Learning: Ein neues Programmierparadigma



Quelle: Vgl. Grafik bei Chollet (2018)

Das Machine Learning – und die darin enthaltene Spezialdisziplin *Deep Learning* – nutzt das Verfahren der *Neuronalen Netze* und wird beispielsweise auch in der Analyse von Literaturtexten eingesetzt.

2.2 IT-Begriffe

Zunächst werden einige IT-Begriffe kurz erläutert und dann die wichtigsten Methoden beschrieben, die für die Literaturtextanalyse genutzt werden, wobei einige der Methoden auch in anderen Anwendungsfeldern gute Ergebnisse liefern, z.B. im Natural Language Processing (NLP). Zum IT-Einsatz einer Methode wird ein geeignetes Softwareprogramm verwendet.

7 URL: https://www.edv-buchversand.de/download/?mode=download&type=chapter&file=itp-838.pdf [Zugriff: 05.11.2023]

8 Chollet 2018, S. 23.

Einige IT-Begriffe:

- Ein *Algorithmus* bezeichnet den internen Ablauf im Softwareprogramm, mit dem eine bestimmte Methode programmtechnisch realisiert wird. Dabei werden die Programmanweisungen in eine bestimmte Reihenfolge gebracht, so dass damit das betreffende Problem gelöst wird (»Rezept«). Ein Algorithmus wird auch als ein »Problemlösungsverfahren mittels einer endlichen Folge von eindeutig bestimmten und tatsächlich durchführbaren Teilhandlungen« (Fischer/Hofer 2008, S. 32) definiert. »Wird ein Algorithmus in eine für Maschinen verständliche Folge von Anweisungen codiert, dann liegt ein Programm vor«. (ebd.)
  Zur Beschreibung des Algorithmus kann Pseudocode, eine Art von vereinfachtem Programmcode, der noch nicht einer bestimmten Programmiersprache zugeordnet ist, verwendet werden (siehe Artikel Spezialthemen im Download-Bereich). Um den Algorithmus zu verifizieren, werden einige Tests durchgeführt, bei denen bestimmte Eingaben die vorher manuell ermittelten Ausgaben mit Hilfe eines Durchlaufs (run) durch den Algorithmus erzeugen sollten.
  Zur Realisierung der Aufgabenstellung ›Zerlegung eines Textes in seine Wortbestandteile‹, kann ein einfacher Algorithmus darin bestehen, dass im Text von vorn beginnend Zeichen für Zeichen nach Worttrennungselementen (WTE) gesucht wird. Ein WTE kann ein Leerzeichen, ein Satzzeichen oder eine Zeilenschaltung sein. Wird ein WTE gefunden, besteht das Wort aus den Zeichen zwischen dem vorletzten WTE (bzw. dem 1. Zeichen) und dem gefundenen WTE (bzw. letzten Zeichen). Damit nicht für jede grammatische Variante des Wortes (Glück, Glücke, Glückes), ein eigenständiges Wort gezählt wird, wird es noch in seine Grundform (Glück) überführt, und mit dem gesuchten Wort verglichen. Ist es gleich, wird ein Zähler für dieses Wort um 1 erhöht (Start bei 0). Am Ende des Textes enthält der Zähler die gesuchte Häufigkeit.
  Als ein Beispiel kann die Häufigkeit des Wortes ›Glück‹ in folgendem Text mit Hilfe des o.g. Algorithmus ermittelt werden: ›Jeder ist seines Glückes Schmied, aber ein Glück kommt selten allein.‹ Die absolute Häufigkeit des Wortes Glück ist 2, die relative ist $2/11 = 0,1818$ (11 ist die Gesamtanzahl der Wörter).
- In der Informatik wird mit *Methode* eine bestimmte Vorgehensweise (Verfahren, Herangehensweise) zur Lösung einer Aufgaben- oder Problemstellung (vgl. Fischer/Hofer 2008, S. 522f.) bezeichnet. Dazu ein einfaches Beispiel: Es soll geprüft werden, ob eine positive ganze Zahl n eine Primzahl ist. Zu dieser Prüfung kann die Methode »Probedivision« angewendet werden, also die sukzessive Division von n durch alle ganzen Zahlen von 2 bis n-1 (bzw. bis zur Wurzel aus n reicht auch). Falls dabei ein Rest = 0 vorkommt, dann ist n keine Primzahl. Diese Methode ist für große n jedoch nicht sehr performant, so dass alternative Methoden genutzt werden sollten, z.B. das ›Sieb des Eratosthenes‹.[9] Im Anhang ist das Beispiel mit Pseudocode und einem Java- und Python-Programm mit Funktion enthalten. Ein anderes Beispiel ist die Methode des Latent Dirichlet Allocation (LDA), die u.a. für das Topic Modeling verwendet wird.

9 URL: https://de.serlo.org/mathe/96712/das-sieb-des-eratosthenes [Zugriff: 04.11.2023]

Im Bereich KI werden für das maschinelle Lernen drei Methoden genutzt, die je nach Aufgabenstellung sinnvoll sind: Überwachtes Lernen mit Trainingsdaten z.B. für Mustererkennung wie Gesichts-, Objekt- oder Spracherkennung, unüberwachtes Lernen z.B. für Clustering und Reinforcement Lernen (Lernen mit try-and-error) z.B. für Brettspiele.[10] Als Methode versteht man hier also eine abstrakte Vorgehensweise, während ein Algorithmus zur Programmierung dieser Methode sehr detailliert ist und u.a. mit Pseudocode beschrieben werden kann (siehe Primzahlbeispiel im Anhang).

- Eine *Funktion* realisiert eine Anforderung der Aufgabenstellung und ist Teil der Problemlösung (Methode). Dazu nutzt man einen bestimmten Algorithmus.
- Ein (Software-)*Programm* enthält eine oder mehrere Funktionen zur Lösung einer oder mehrerer vorgegebenen/r Anforderung/en (auch Aufgaben). Eine Anforderung kann z.B. sein, dass die Häufigkeit eines vorgegebenen Wortes in einem Text oder einer Textsammlung (Korpus) ermittelt werden soll. Dazu muss der Text erstmal in seine Wortbestandteile aufgespalten werden (*Tokenisierung*). Ebenso sollte das Wort unabhängig von Groß- und Kleinschreibung sowie von Kasus und Numerus sein; bei der *Lemmatisierung* werden zum Beispiel die Wortformen Glück, Glücks und Glücke auf seine Grundform, also Glück, zurückgeführt, da dann nur die Grundform und nicht alle Varianten des Worts gezählt werden.
- Softwareprogramme müssen mit Hilfe einer *Programmiersprache* entwickelt (programmiert) werden. Welche Programmiersprache verwendet wird, hängt von einigen Faktoren ab: für welche Plattform(en) (Windows, Unix/Linux, iOS, Android, Web) soll der ablauffähige Code erzeugt werden, gibt es große Bibliotheken (fertige Programmeinheiten, die eingebaut werden können), welche Sprache(n) beherrscht/en der/die Programmierer:innen (Python, System R, Java, C#, JavaScript, VBA, C++, …). Nach der Programmierung mit Hilfe einer Entwicklungsumgebung (z.B. RStudio) liegt das Programm im Quellcode vor. Nun muss es noch für den Zielcomputer, auf dem es laufen soll, übersetzt werden. Dazu wird der Quellcode bei einigen Programmiersprachen in einen Zwischencode übersetzt, der beim Ablauf von einem Interpreter in den Maschinencode umgesetzt wird. Dieser Maschinencode wird von dem Prozessor/der CPU des Zielcomputers verstanden und kann abgearbeitet werden; er wird dabei speziell für diesen Prozessor vom Interpreter generiert. Statt des Zwischencodes kann auch Maschinencode mit Hilfe eines Übersetzers (Compilers) erzeugt werden, der dann ohne Interpreter direkt auf dem Prozessor des Zielsystems ausgeführt werden kann. Solche für den Prozessor compilierten Programme (z.B. in C programmiert) sind deutlich schneller als Programme, die noch zur Laufzeit interpretiert werden müssen. Der Compiler ist in dem Fall auf die Sprache und den Zielprozessor spezialisiert. Für die Frage, welche Programmiersprache verwendet werden soll, gilt damit auch das Kriterium, auf welcher Plattform/CPU das Programm (mittels Compiler oder Interpreter) ablaufen kann. Softwareprogramme können heruntergeladen, lokal installiert und dann ausgeführt oder mithilfe eines Webbrowsers genutzt werden; häufig handelt es sich dabei um Cloud-Lösungen.

10 Vgl. den Beitrag von A. de Vries im vorliegenden Band.

- *Preprocessing*/Vorverarbeitung: Bevor die eigentliche Textanalyse startet, wird der Text passend aufbereitet, z.B. in Abschnitte oder Wörter zerlegt (Parsing, Tokenisierung) oderder Vorarbeiten an Texten zum Einsatz von KI-Tools wie Aufteilung in Trainings- und Testdaten.
- In einer *Pipeline* werden mehrere Funktionen eines Tools hintereinandergeschaltet, um sie nacheinander auf einer Datenmenge, z.B. einem Text, auszuführen. Dadurch muss der/die Nutzer:in die Funktionen nicht mehr einzeln nacheinander aufrufen.
- Falls eine Programmiersprache *o*bjektorientierte *Programmierung* (OOP) unterstützt, können Objekte (z.B. Personen) auf Basis einer (Personen-)Klassendefinition erstellt und verwaltet werden. Damit wird die Programmierung deutlich komfortabler und sicherer. Die funktionale Programmierung ist die klassische Form und wird auch *prozedurale* Programmierung genannt. Die *aspektorientierte* Programmierung unterstützt eine stärkere Modularisierung, was die Wartbarkeit (Erweiterbarkeit) des Programms erhöht.
- *Künstliche Intelligenz* ist ein Sammelbegriff für Software, die die natürliche Intelligenz nachahmt. Eine der Vorgehensweisen dabei ist, dass die Software mithilfe von Trainingsdaten, also aus bekannten Ergebnissen oder Ist-Werten, lernt und dabei ein neuronales Netz aufbaut. Wenn eine KI-Software z.B. bei einem Bild unterscheiden soll, ob es sich um ein Auto oder ein Zweirad handelt, wird man diese Software mit vielen Bildern von Autos und Zweirädern trainieren (Trainingsphase). Nachdem die Software ausreichend anhand von Trainingsdaten (ca. 70 % der zur Verfügung stehenden Ergebnis-Daten) trainiert bzw. gelernt hat, wird man das Gelernte noch anhand von Testdaten, d.h. mit weiteren Bildern (in der Regel mit den übrigen 30 % der zur Verfügung stehenden Datenmenge) überprüfen. Wird das Gefährt in der Testphase auf dem Bild mit einer hohen Trefferrate richtig erkannt (z.B. in über 95 %), ist das Ergebnis zufriedenstellend. In anderen Beiträgen des vorliegenden Buches wird näher auf KI eingegangen.
- Ein *Chat-Bot* ist eine Anwendung, die Künstliche Intelligenz verwendet, um sich mit Menschen in natürlicher Sprache (Natural Language) zu unterhalten. Benutzer können Fragen stellen, auf die der Chat-Bot in ganzen Sätzen antwortet. Aktuell sind Chancen und Risiken des leistungsfähigen Chat-Bots ChatGPT (Generative Pre-trained Transformer) Gegenstand von öffentlichen Diskussionen.

2.3 IT-Einsatz in der Literaturtextanalyse

Die Kernaufgabe der Informatik ist die Durchführung einer Problemlösung/Aufgabenstellung mithilfe eines Computers. Dabei ist Computer im weitesten Sinn zu verstehen, wobei dazu Großrechner, Server, Clouds, PC's, Laptops, Notebooks und Handys usw. gehören. Gegenüber einem Menschen kann ein Computer diese Problemlösung viel schneller bearbeiten, macht weniger Fehler dabei (vorausgesetzt, das Programm ist ausreichend getestet) und ist in der Lage, sehr große Datenmengen (big data) einzubeziehen. Wenn z.B. eine Korpusanalyse aller deutschsprachigen Romane des 18. Jahrhundert ansteht, würde ein Mensch dazu allein zum Lesen schon sehr lange Zeit benötigen. Deshalb bleibt vieles in dem Bereich des *Great Unread* und kann erst durch Einsatz von digitalen Methoden (wie in den Digital Humanities) erschlossen werden.

Für die Lösung vieler Aufgabenstellungen in der Literaturtextanalyse werden eine Reihe von Programmen/Tools eingesetzt u.a. CATMA, spaCy oder andere, siehe weiter unten. Bei diesen Programmen kann der/die Anwender:in die Texte in einem passenden Format in das Tool importieren; Dann stehen ihm/ihr die Funktionen zur Textanalyse i.d.R mit grafischer Benutzeroberfläche (GUI Graphical User Interface) und grafischen Anzeigen der Ergebnisse zur Verfügung. Die Programme können entweder heruntergeladen und lokal installiert ober mithilfe eines Webbrowsers genutzt werden.

Häufig werden auch Tools eingesetzt, bei denen man noch selbst programmieren muss, z.B. das System R inkl. der GUI RStudio oder die Programmiersprachen Python oder Java; das Tool ist dann die betreffende Entwicklungsumgebung inkl. Editor. Bei diesen Tools stehen Bibliotheken (sogen. packages, libraries) mit speziellen Funktionen für die Bearbeitung von Literaturtexten zur Verfügung, bei dem System R die Bibliothek stylo für Stilometrie oder topicmodels für Topic Modeling, bei Python Funktionen wie split_into_words (Text in Wörter aufteilen) oder Counter (Häufigkeit eines Wortes). Mit diesen Zusätzen ist der noch zu leistende Programmieraufwand für die Analyse deutlich reduziert.

2.4 Textformate und -codierung

Ein Literaturtext, der mithilfe von Softwaretools analysiert werden soll, muss in digitaler Form vorliegen. Neben txt- (reiner Text) oder csv- (Character Separated Values) wird das XML- (eXtended Markup Language) und das darauf basierende TEI-Format (Text Encoding Initiative) genutzt.

Die TEI ist ein Konsortium, das einen Standard für die Darstellung, den Austausch und die Speicherung von Texten in digitaler Form entwickelt hat.[11] Dieser Standard (TEI-Kodierung) basiert auf XML und definiert ein Dokumentenformat in Form einer Metasprache. Es ist zu einem De-facto-Standard in den Geisteswissenschaften geworden und liegt aktuell in der Version P5 vor.[12]

TEI enthält mehrere Module, die bestimme Elemente zur Festlegung der Dokumentstruktur, zur Markierung von Zeilen und Seiten, Tabellen, Anmerkungen, zur Auszeichnung von Gedichten und Dramen und anderes mehr enthalten.[13]

Die Elemente sind mit sogenannten Tags gekennzeichnet. Diese sind in spitzen Klammern <…> eingeschlossen (Markup), wobei es jeweils ein Start- und ein Ende-Tag (letzteres beginnt mit/) gibt.
Beispiel für Tags in dem Drama Schnitzlers Reigen:

```
<listPerson>
<person xml:id="soldat" sex="MALE">
<persName>Der Soldat</persName>
</person>
```

11 Vgl. URL: https://tei-c.org/ [Zugriff: 04.11.2023].

12 Vgl. ebd.

13 Vg. Ebd.

```
… weitere Personen
</listPerson>
```

Mit dem Tag <listPerson> wird die Liste der Personen/Figuren gestartet und mit </listPerson> beendet. Dazwischen wird jede Person mit einem id-Tag und Name-Tag definiert.
Die einzelnen Szenen werden wie folgt definiert, hier für Szene **I**:

```
<div type="scene">
<head>
<pb n="327"/> I</head>
```

Bühnenanweisungen werden mit

```
<stage>Spät abends. An der Augartenbrücke.</stage>
```

gekennzeichnet.
Für gesprochene Texte wird folgendes verwendet:

```
<sp who="#soldat">
<speaker>SOLDAT.</speaker>
<p>Ah, ich bin der schöne Engel?</p>
</sp>
```

Mit diesen Tags werden bestimmte Elemente gekennzeichnet, die für Textanalysen genutzt werden können, z.B. hier die spielenden Figuren mit ihren Texten in den verschiedenen Szenen. Literaturtexte werden auch im Standard-XML- oder einfachen txt-Format zum Download zur Verfügung gestellt.

Bei der Codierung spielt auch der Zeichencode eine wichtige Rolle. Neben dem ASCII- sowie dem Windows-Zeichencode wurde ein UTF-8 Zeichencode eingeführt, der als Unicode bezeichnet wird. UTF-8 ist in den ersten 128 Zeichen identisch mit dem ASC II-Code, hat meist nur ein Byte Speicherbedarf und eignet sich für Zeichen der meisten westlichen Sprachen, wodurch UTF-8 zu einer De-facto-Standard-Zeichenkodierung für das Internet geworden ist. Eine UTF-8-Codetabelle ist in https://www.utf8-zeichentabelle.de/ zu finden.

2.5 Praktische Vorgehensweisen

Um Texte zu analysieren, benötigt man einen oder mehrere Texte und geeignete Tools für die Analyse. Texte sind in vielen Quellen im Internet zu finden, hier einige mit ihren Web-Adressen.

- Drama Corpus
  »Der German Drama Corpus (GerDraCor) enthält ca. 500 deutschsprachige Stücke. Die Texte sind TEI P5-kodierte und somit die Szeneneinteilung reproduzierende

XML-Dokumente.« (Fischer u.a. 2022). Die Internet-Adressen (URL) dazu sind dracor.org (international) und gerdracor.de (deutscher Raum).
- Deutsches Textarchiv (DTA), verschiedene Textformate
  https://www.deutschestextarchiv.de/
- TextGrid Repository
  https://textgrid.de/ mit Tools, Texten und Tutorials
  https://textgridrep.org/ zum Download von Texten im xml-Format
- DigBib.Org, Die digitale Bibliothek; Texte im pdf-FormatKOLIMO
  https://kolimo.uni-goettingen.de/index.html
- Gutenberg-Projekt
  https://www.projekt-gutenberg.org/
  Die Texte können häufig nur kapitelweise heruntergeladen und z.B. in einer txt-Datei gespeichert werden.
- Zeno.org
  https://www.zeno.org/

Wenn ein Analyseprogramm txt-Dateien verarbeiten kann, können diese direkt in dem txt-Format heruntergeladen werden. XML-Dateien können nach dem Download mit geeigneten Tools in txt-Dateien umgewandelt werden, z.B. mit Microsoft Word© oder einem XML-Editor. Bei Word wird die XML-Datei geöffnet und mit »Speichern unter« als »Nur Text« in einer txt-Datei gespeichert. Dabei sollte die Einstellung »Textcodierung nach Windows (Standard)« und »Zeilenumbrüche einfügen« gewählt werden. PDF-Dateiinhalte werden in eine txt-Datei kopiert. Bei Bedarf kann Word Texte auch im xml-Format speichern.

Die Analysetools sind oft im Web zu finden, meist kostenfrei als sogen. »Open Source« Software. Sie können teilweise direkt als Web-Applikation genutzt werden oder müssen auf einen lokalen PC oder Server heruntergeladen werden. Im Fall von ausführbaren exe-Dateien müssen sie dann installiert und evtl. noch konfiguriert werden. Im Fall von Source-Code (Softwareprogramm) muss das passende Ausführungssystem lokal installiert werden, z.B.

- bei einem Java-Programm (ist meist eine jar-Datei) eine JRE (Java Runtime Environment) oder, wenn man auch noch programmieren möchte, eine JDK (Java Development Kit) plus Editor (z.B. Java-Editor oder Eclipse); teilweise müssen noch benötigte Erweiterungen heruntergeladen und installiert werden,
- bei einem Python-Programm eine IDE (Integrated Development Environment), z.B. IDLE (Integrated Development and Learning Environment) plus Anaconda-Editor oder Visual Studio Code; auch hier sind häufig Erweiterungen (libraries) notwendig,
- bei einem System R Skript das System R selbst und als Editor RStudio; Erweiterungen werden als packages heruntergeladen und installiert, z.B. stylo für die Stilometrie, siehe dort.

Auf die betreffenden URLs wird hier verzichtet, da sie sehr leicht im Internet zu finden sind.

2.6 Statistische Merkmale eines literarischen Textes

In diesem und den folgenden Abschnitten werden nur einige der oben aufgelisteten digitalen Methoden etwas näher beschrieben und erläutert.

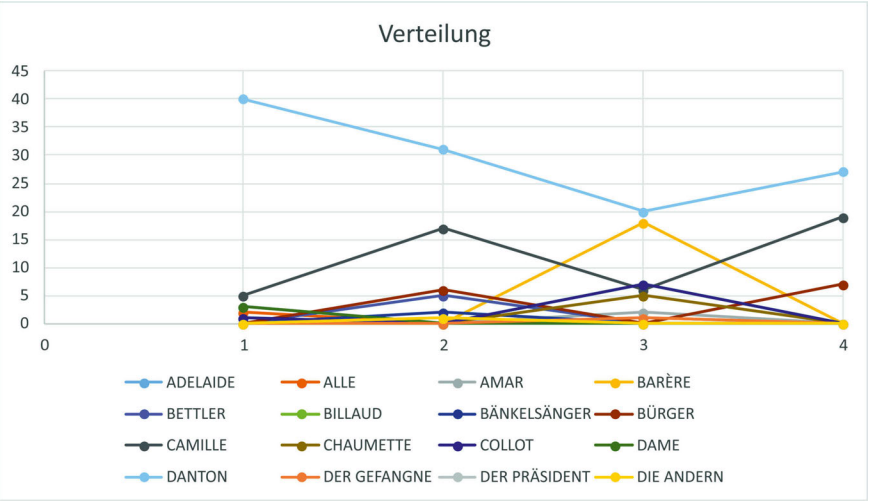
- Die statistischen Merkmale eines Textes werden bei computationellen Analysen literarischer Texte als erstes ermittelt; dazu gehören u.a.: Anzahl der Zeilen, Zeichen, Wörter und Schlüsselwörter
- Häufigkeit aller Wörter (mit/ohne Stoppwörter); die Häufigkeiten können grafisch mit Wordclouds angezeigt werden, siehe Abbildung 2.
- Nachbar-Wörter (Kontext, KWIC = Keywords in Context) von Schlüsselwörtern mit Vorgabe der Anzahl der Nachbarn, sowie deren Häufigkeiten. Grafische Visualisierung mit DoubleTrees.
- Verteilung der Schlüsselwörter in (gleich langen) Textsegmenten mit Distributionsgraph, siehe Abbildung 3.
- Die Verteilung der Figuren bei Dramen: Welche Figur hat wie viele Redebeiträge in den Szenen und welche Figur tritt mit welcher Figur in den Szenen gemeinsam auf, siehe Konfigurations- und Adjazenzmatrix in Kap. 3.5
- Annotationen einzeln oder im Team mit frei definierbaren Annotationskategorien
- Manuelle Queries (Abfragen), zum Beispiel wie in CATMA, siehe Kap. 4.1:
  - Wie viele Wörter enthält der Text?
  - Was sind die am häufigsten vorkommenden Inhaltswörter, d.h. Wörter mit »mehr« Bedeutung als Funktionswörter wie Artikel, Pronomen usw.?
  - Wie viel Wörter beginnen mit »A«?
  - Welche Wörter kommen mehr als 5-mal vor?

Im Folgenden werden Abbildungen zur Erläuterung der oben genannten statistischen Merkmale gezeigt:

Abb. 2: Wordcloud zu Büchners »Dantons Tod«

Quelle: Mit Excel© erstellte Grafik aus der Ausgabe von *LitAs, siehe Kap. 4.6*

Abb. 3: Verteilung bei Dramen: Welche Figur tritt in welchem Akt (1–4) in Büchners Dantons Tod wie oft auf (hier Anzeige der 16 Figuren mit den häufigsten Auftritten) x-Achse: 4 Akte; y-Achse: Häufigkeiten.



Quelle: Mit Excel© erstellte Grafik aus der Ausgabe von LitAs, siehe Kap. 4.6

3 Digitale Methoden

3.1 Digitale Methoden und Verfahren für die Literaturtextanalyse: Basis-Begriffe

Folgende digitale Methoden werden in der computationellen Literaturtextanalyse eingesetzt; viele kommen aus der Sprachverarbeitung (NLP) (hier nur einige, in alphabetischer Reihenfolge):

- *Annotation* (siehe Kap.3.2).
- *Distant Reading* und *Close Reading* bezeichnen Ansätze aus den digitalen Literaturwissenschaften, die in Kap. 3.3 näher beschrieben werden.
- Die *EDA* (Explorative Datenanalyse) entspricht einer induktiv-explorativen Vorgehensweise, bei welcher nach Auffälligkeiten oder Trends in Datensätzen, Texten oder Korpora gesucht wird. Für explorative Ansätze kommen prinzipiell viele der nachfolgend genannten Tools in Frage.
- *Häufigkeitsverteilung* ist die Verteilung der Häufigkeit eines Wortes in einem Text, der durch gleich lange Abschnitte, Kapitel oder Szenen o.ä. definiert sind.
- *Lemmatisierung* bezeichnet die Reduktion der Wortform auf ihre Grundform.
- *Named Entity Recognition* (NER, Erkennung von Eigennamen) ist eine Methode, bei der eindeutig benannte Größen (Entitäten) automatisch vom Programm erkannt und im Text markiert werden. Typische Beispiele für NER-Kategorien sind Personen, Länder, Orte, Produkte, Organisationen oder Buchtitel.

- *Netzwerkanalyse (s. auch: Konfigurations- und Adjazenzmatrix*, Kap. 3.5): Ein Netzwerk besteht aus Knoten und Kanten. Damit kann man bestimmte Zusammenhänge visualisieren, wobei einerseits die Topologie des Netzes Informationen liefert, andererseits kann man auch mit Dicke und Farbe Informationen auf die Knoten und Kanten legen. Beispiele für solche Netze sind die Figurenkonstellationen in einem Drama (wer tritt mit wem in welcher Szene auf) oder die Distanzen von Literaturtexten in der Stilometrie (z.B. das Simmelian Backbone Network, vgl. Weitin 2021).
- *KI-basiertes Clustering* (siehe Kapitel 3.4).
- *Parsing* bezeichnet die Erkennung der Konstituenten eines Wortes, Satzes bzw. Textes. Mit einem Parser werden grundsätzlich (Programm-)Eingaben zur Weiterverarbeitung überprüft und geeignet angepasst. Beispielsweise ist der Parser in einem Computersprache-Übersetzer (Compiler) für die syntaktische Prüfung des Programms zuständig.
- *POS-Tagging* (Part-of-Speech-Tagging) ist die Zuordnung von Wörtern eines Textes zu Wortarten, siehe dazu auch das Python-Programm in dem Kapitel 4.3 zu spaCy weiter unten. Für das POS-Tagging wird neben Verfahren des NLP auch KI eingesetzt, um Vorhersagen darüber zu treffen, welches Tag oder Label in diesem Kontext am ehesten zutrifft. Z. B. ist ein Wort, das in einem deutschen Text auf der/die/das folgt, i.d.R. ein Substantiv.
- Die *Sentimentanalyse* ist ein Teilgebiet der Literaturtextanalyse und bezeichnet die gezielte Suche von bestimmen Informationen in Literaturtexten. Dabei liegt der Fokus der Suche auf Gefühlen und Empfindungen, aber auch auf individuellen Meinungen, die in den Texten erkannt und ausgewertet werden (vgl. Flüh 2019).
- Ermittlung von *statistischen Merkmalen* wie die Häufigkeit von (bestimmten) Wörtern (MFW: Most Frequent Words), z.B. Schlüsselwörtern oder Nachbarn (KWIC: Keywords in Context).
- *Stilometrie* (siehe Kap. 3.6).
- Verwendung von Stoppwortlisten: Bestimmte Wörter werden bei der Ermittlung von Worthäufigkeiten ausgeblendet, z.B. Wörter wie *und*, *oder*, *damit* usw. Es handelt sich dabei meist um Artikel, Konjunktionen oder Pronomen, die in Stoppwortlisten gespeichert und bei den statistischen Merkmalen wie Worthäufigkeit nicht mitgezählt werden.
- *Textdigitalisierung* bezeichnet die Umwandlung eines handschriftlichen oder gedruckten Textes in einen digitalen Text (wie durch OCR: Optical Character Recognition). Tools dazu sind u.a. Abbyy FineReader (kommerzielle OCR-Software) oder OCR4all als kostenfreies OCR-Tool, siehe Kap. 4. Die OCR-Schrifterkennung sollte nur dann eingesetzt werden, wenn die Qualität des zu digitalisierenden Textes gut ist, sonst ist der Aufwand für die Nacharbeit zu hoch. Eine Alternative besteht darin, dass der Text einer Spracherkennungssoftware vorgelesen wird. Bei kleineren Texten ist kann der Nachbearbeitungsaufwand dann meist geringer als bei OCR.
- *Tf-idf* (term frequency – inverted document frequency) ist ein Maß für die Bedeutung eines Wortes/Terms hinsichtlich seiner Fähigkeit, Dokumente voneinander zu unterscheiden (Ähnlichkeit). tf-idf = term_frequency * inverted_document_frequency mit term_frequency = Anzahl der Vorkommen eines bestimmten Terms im Do-

kument inverse_document_frequency = log (Gesamtzahl der Dokumente/Anzahl der Dokumente mit Term) + 1; log zur Basis 2

- *Tokenisierung* bezeichnet in der Computerlinguistik die Vorverarbeitung eines Textes mittels Segmentierung in Einheiten, z.B. der Wortebene. Erst danach kann die Textanalyse auf Basis der Tokenisierung durchgeführt werden. Zur Aufteilung eines Satzes in Wörter kann als einfacher Algorithmus die Abtrennung nach jedem Leer- und Satzzeichen verwendet werden. In vielen Tools sind dazu entsprechende Bibliotheken enthalten (Python, NLTK, spaCy, Deras, Gensim).
- *Topic Modeling* (siehe Kapitel 3.7).
- Mit *Visualisierung* wird ein Sachverhalt, etwa das Ergebnis einer Analyse, visuell z.B. mittels Grafiken (Netzwerkgraphen) oder farbig abgestuften Tabellen sichtbar gemacht. »Visuelle Darstellungen können neue Einsichten in Textdaten und deren innere Zusammenhänge liefern. Textvisualisierungen unterstützen sowohl die Kommunikation von Forschungsergebnissen als auch die explorative Analysetätigkeit.«[14] Eine *Wordcloud* zeigt eine Wolke mit Wörtern, wobei die Häufigkeit innerhalb des semantischen Feldes (wie etwa bei Topic Modeling, vgl. Weitin 2021, S. 119) durch die graphische Größe des Wortes dargestellt wird und das häufigste Wort in der Mitte steht.
- Mit *Worteinbettung* (*Word Embedding*) und Vektorisierung werden die Wörter eines Textes in numerische Vektoren umgewandelt; auf Basis dieser Vektoren können Zusammenhänge sowie der Kontext der Wörter zueinander mathematisch berechnet werden. Ein Tool für Worteinbettung ist Word2vec, das auf einem neuronalen Netz basiert (vgl. Thamm 2023).
- *Zipfsches Gesetz* (vgl. Leopold 2022) ist ein Verfahren und besagt, dass die Häufigkeit eines Wortes (Events) umgekehrt proportional zu seiner Position in der Reihenfolge ist. Es gilt damit (in fast jeder Sprache), dass das häufigste Wort etwa doppelt so häufig vorkommt wie das zweithäufigste, das dritthäufigste Wort ein Drittel so oft wie das häufigste usw. Damit ist die Wahrscheinlichkeit p des Auftretens eines Wortes umgekehrt proportional zu seinem Platz n (Rang) auf der Häufigkeitsliste: $p(n) \sim 1/n$.[15]

3.2 Annotation

Unter Annotation versteht man die manuelle oder automatische Ergänzung von zusätzlichen Informationen zu einem Text, z.B. Kommentare, Anmerkungen, Zuweisen von Analysekategorien usw. Das können Hervorhebungen oder auch Textergänzungen sein. Als gelungene digitale Umsetzung von hermeneutischen Verfahrensweisen zeigt sich das »undogmatische«, d.h. dynamisch erweiterbare, textanalytische Tool CATMA[16]: Hier sind »freie Annotationen nach individuell definierten Kategorien, Mehrfachannotation

14 URL: https://fortext.net/routinen/methoden/textvisualisierung [Zugriff 06.11.2023]

15 Vgl. URL: https://bernardzitzer.com/de/zipfsche-gesetz-zipfs-law/ [Zugriff: 05.11.2023]

16 Vgl. URL: https://fortext.net/tools/tools/catma [Zugriff: 08.11.2023]

einzelner Wörter und Passagen, überlappende Annotation [oder] widersprüchliche Annotation«[17] möglich.

Während die manuelle Annotation händisch vorgenommen wird, kann eine (teil-)automatisierte Annotation u.a. durch Machine-Learning-Verfahren durchgeführt werden kann (vgl. Jacke 2018). Ein Goldstandard wird auf der Basis der intersubjektiven Übereinstimmung von kollaborativen Annotationen erstellt (vgl. Gius 2015). Mit einem Python-Package GitMA kann direkt auf das Backend von CATMA zugegriffen werden, um Annotationsdaten (in CATMA) in Tabellen zu visualisieren und einen Goldstandard zu erstellen (vgl. Schumacher/Vauth 2022, § 1). Im Abschnitt über CATMA (siehe Kap. 4.1) wird näher auf das Thema eingegangen.

3.3 Distant Reading und Close Reading

Wegen des großen Aufwands ist es für einen Menschen oder auch eine Gruppe nahezu unmöglich, sehr große Textmengen zwecks Analyse dieser Texte zu lesen. Mit Computern ist das aber kein Problem mehr, da die Lese- und Verarbeitungsgeschwindigkeit drastisch steigt. Der Begriff Distant Reading geht auf Franco Moretti zurück und bedeutet in diesem Zusammenhang, dass »computationelle Verfahren auf viele Mengen von Textdaten angewandt werden, ohne dass die Texte selber gelesen werden.«[18] Dabei werden quantitative Analyseverfahren eingesetzt, wobei auch qualitative Metadaten quantitativ untersucht und verglichen werden können.

Der Gegenbegriff zu Distant Reading ist Close Reading. Dabei werden die Texte sorgfältig gelesen und analysiert bzw. interpretiert, wobei die manuelle Annotation eine wichtige Rolle spielt. Inzwischen sind digitale Analyseprogramme wie z.B. CATMA dazu in der Lage, die Annotationen zumindest teilweise automatisiert durchzuführen.

3.4 KI-basiertes Clustering

In der Literaturtextanalyse wird KI (Künstliche Intelligenz) seit vielen Jahren eingesetzt. Das liegt unter anderem daran, dass immer mehr Daten (Big Data) für das Trainieren (überwachtes Lernen) von KI-Tools vorliegen und dass KI immer leistungsfähiger bzgl. der Hardwarebasis und der Softwaretechnik wird. KI hat gegenüber einfachen multimodalen Analysetechniken den Vorteil, dass die Einflussparameter beim Trainieren automatisch erkannt werden und nicht mühsam manuell ermittelt werden müssen. Ein Anwendungsbereich für KI ist das Clustering, d.h. Einordnung von Texten in Clustern, wozu bestimmte Textattribute (z.B. Gattung, Epoche, Gender, ggfs. Kanonizität oder Bekanntheitsgrad usw.) genutzt werden können. Weitere Anwendungsbereiche der KI sind Annotationen, Sprachanalysen (Parts-of-Speech-Analyse, Syntaxanalyse/Parsing), Sprachübersetzung, Sentimentanalyse, Sprach- und Befehlserkennung sowie ChatBots. In weiteren Beiträgen des vorliegenden Bands werden KI-Systeme im Allgemeinen sowie ihre Möglichkeiten und Grenzen detailliert beschrieben, sodass das Thema in

17 Ebd.

18 Distant Reading Glossar. In: forTEXT. Literatur digital erforschen. URL: https://fortext.net/ressourcen/glossar [Zugriff: 07. 08. 2023]

diesem Kapitel nicht weiter vertieft wird. Auf dem Download Server wird ein Beispiel für die KI-basierte Methode das Clustering von Autorendaten entwickelt.[19]

3.5 Netzwerkanalyse: Konfigurations- und Adjazenzmatrix bei Dramen[20]

Bei der Analyse von Dramen sind neben den statistischen Auswertungen auch folgende Fragestellungen von Interesse:

- Wie häufig tritt eine Figur in welcher Szene auf?
- Welche Figuren agieren mit welchen zusammen auf der Bühne?
- Wie umfangreich sind dabei die Redebeiträge der Figuren?

Zu diesem Zweck werden Netzwerkanalysen erstellt und mit Hilfe von Konfigurations- und Adjazenzmatrizen (vgl. Lück (o.J.), S. 243ff., vgl. auch Pfister 2004, S. 44) visualisiert.

Der rumänische Mathematiker und Philosoph Solomon Marcus hat sich bereits ab den 1960er Jahren mit diesem Thema auseinandergesetzt und dazu mathematische Konzepte entwickelt (vgl. Marcus 1973, S. 287ff, vgl. Lück o.J., S. 243). Unter anderem hat er die sogenannte *Konfigurationsmatrix* konzipiert, eine zweidimensionale Tabelle, die in der i-ten Zeile (z.B. in der Zeile 7, Figur HERMANN) und der j-ten Spalte (z.B. j = 4, Szene 2.1) die Anzahl der Redebeiträge der Figur in dieser Szene angibt (hier 21).

Tabelle 1: Beispiel einer Konfigurationsmatrix (Ausschnitt aus Schillers Die Räuber), Häufigkeit der Auftritte/Redebeiträge der Figuren in den einzelnen Szenen

Szenen	1.1	1.2	1.3	2.1	2.2	2.3	3.1	3.2	4.1	4.2	4.3	4.4	4.5	5.1	5.2	Sum
AMALIA			20		33		20			11		14			13	111
BEDIENTER														5		5
DANIEL					1					25	15			26		67
DER ALTE MOOR	28				38								8		20	94
FRANZ VON MOOR	29		22	22	16		9			27				47		172
GRIMM		7						10					4	3	5	29
HERMANN				**21**	11		10						10			52
KARL VON MOOR		29				25		37	2	12	20	11	31		44	211
KOSINSKY								21	1		4					26
MOSER														17		17
PATER						13										13

Quelle: mit Excel© aufbereitete Tabelle aus der Ausgabe von LitAs (siehe Kap. 4.6)

19 https://github.com/LuckeJohannes/Literaturwissenschaft_Informatik

20 Vgl. URL: https://dracor.org/ [Zugriff: 15.05.2023]

Marcus nennt dabei zwei Figuren

»**szenisch konkomitant**, wenn sie stets zusammen auf der Bühne sind
szenisch dominant, wenn die eine nur dann auftritt, wenn auch die andere – dominante – Figur auf der Bühne ist […]
szenisch unabhängig, wenn zwischen ihnen weder ein szenisch konkomitant noch ein szenisch dominantes Verhältnis besteht.
Die szenische Unabhängigkeit kennt zwei Sonderfälle: Zwei Figuren sind
szenisch alternativ, wenn sie in keiner Szene gemeinsam auftreten
szenisch komplementär, wenn sie szenisch alternativ sind und die Vereinigungsmenge der Szenen, in denen jede anwesend ist, gleich der Menge aller Szenen des Stückes ist.«[21]

Auf dieser Basis können »Beziehungen« zwischen Figuren und eine soziale Netzwerkanalyse angeschlossen werden (vgl. ebd.).

Eine Variante der Konfigurationsmatrix besteht darin, dass statt der Anzahl der Redebeiträge nur eine »1« in der i-ten Zeile und der j-ten Spalte enthalten ist, wenn die Figur i mindestens 1x in der Szene ij auftritt. In dem Fall wird die Matrix als Binärmatrix bezeichnet, da nur 0 und 1 vorhanden sind und beschreibt quasi die Szenenstruktur des Dramas.

Auf Basis der Konfigurationsmatrix kann ein Netzwerkgraph als »Beziehungsgraph« zwischen den Figuren eines Dramas ermittelt werden. Der »Beziehungsgraph« zeigt grafisch an, wie oft je 2 Figuren gemeinsam in den Szenen auftreten. Dazu wird eine Adjazenzmatrix (n × n-Matrix, Tabelle) A= (aij) gebildet, die wie folgt definiert ist: Ein Eintrag aij enthält für ein Paar zweier Figuren des Stückes die Anzahl der gemeinsamen Auftritte der beiden Figuren in allen Szenen (vgl. Lück o.J., S. 249). Z.B. treten HERMANN und AMALIA gemeinsam in den Szenen 2.2 und 3.1 auf, so dass der Eintrag aij = a71 = 2 ist (Zeile HERMANN und Spalte AMALIA), siehe 4.

Zusätzlich kann die Anzahl der Redebeiträge (*turn takings*), die die Figuren in einer gemeinsamen Szene haben, ›in die Rechnung eingehen‹ (vgl. ebd., S. 254); dabei kann z.B. für jede gemeinsame Szene das Minimum (der Anzahl Redebeiträge pro Figur) oder das Maximum oder jeweils die Summe beider Werte aus den Szenen aufaddiert werden (vgl. ebd.). »Durch die Verwendung des Minimums wird der Abstand zwischen Haupt- und Nebenfiguren in den Kantengewichten deutlicher« (ebd.), mit dem Maximum werden Nebenfiguren zu dominanten Figuren erhoben (vgl. ebd.). Mit Hilfe des jeweils größten Eintrags in der Matrix kann eine Normierung durchgeführt werden (vgl. ebd.). Eine weitere Alternative für die Gewichte ist, die Beitragsanzahl durch die Beitragslänge in Wörtern zu ersetzen, um die Bedeutung der Figur noch besser zu charakterisieren (vgl. ebd.).

Mit einer Spalte »DEGREE« in der Adjazenzmatrix, die die Anzahl der Kanten, die an einen Knoten grenzen, angibt, kann der Mittelwert der Grade aller Figuren des Stückes angezeigt werden (vgl. ebd.). Bei diesem Graphen sind die Kanten nicht gerichtet, da

21 Marcus 1973, S. 287ff.

es sich bei der Adjazenzmatrix um eine obere Dreiecksmatrix handelt; das obere rechte Dreieck hat dieselben Werte wie das untere linke.

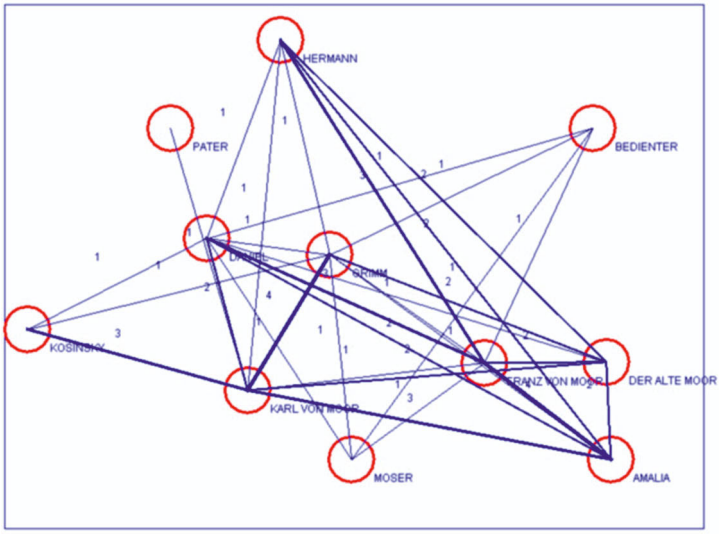
Während in der Konfigurationsmatrix noch die Zeitabfolge in Form der Szenensequenz dargestellt wird, ist die Zeitabfolge in der Adjazenzmatrix nicht mehr enthalten.

Abb. 4: Beispiel einer Adjazenzmatrix zu der obigen Konfigurationsmatrix: wie oft tritt eine Figur mit einer anderen Figur in den Szenen auf

Adjazenzmatrix	*AMALIA*	*BEDIENTER*	*DANIEL*	*DER ALTE*	*FRANZ*	*GRIMM*	*HERMANN*	*KARL*	*KOSINSKY*	*MOSER*	*PATER*	***DEGREE***
AMALIA			2	2	4	1	2	3				**14**
BEDIENTER			1		1	1				1		**4**
DANIEL	2	1		1	3	1	1	2	1	1		**13**
DER ALTE MOOR	2		1		2	2	2	2				**11**
FRANZ VON MOOR	4	1	3	2		1	3	1		1		**16**
GRIMM	1	1	1	2	1		1	4	1	1		**13**
HERMANN	2		1	2	3	1		1				**10**
KARL VON MOOR	3		2	2	1	4	1		3		1	**17**
KOSINSKY			1			1		3				**5**
MOSER		1	1		1	1						**4**
PATER								1				**1**

Quelle: Mit Excel© erstellte Grafik aus der Ausgabe von LitAs

Abb. 5: Netzwerkgraph zur obigen Adjazenzmatrix



Quelle: Visualisierung mit LitAs

Die Werte in der Adjazenzmatrix (Kantengewichte) können noch **normiert** werden, damit die Beziehungen eines Theaterstücks mit denen eines anderen vergleichbar werden. Als Normierungsquotient kann man z.B. den größten Wert in der Adjazenzmatrix nehmen und alle Werte durch diesen dividieren. Dann sind alle Werte zwischen 0 und 1.

Normierte Werte können auch besser visualisiert werden. In der Visualisierung wird dadurch diejenige Kante, die die numerisch stärkste Beziehung zweier Figuren darstellt, für jedes Stück gleich breit gezeichnet.

Die Adjazenzmatrix kann auch grafisch visualisiert werden (Netzwerkgraph), indem die Figuren die Knoten und die Matrixinhalte aij die Kanten (Beziehungen, gemeinsame Auftritte) bilden, siehe 5. Dabei zeigt die Dicke der Verbindungslinie zwischen zwei Figuren die Anzahl der Beziehungen, die zusätzlich auch als Zahl an der Kante angezeigt wird. Mit dieser Grafik lässt sich gut erkennen, welche die zentralen Figuren sind, da jede Kante des Graphen als Gewicht die gemeinsamen Auftritte erhält (vgl. ebd., S. 249).

3.6 Stilometrie

Eine in den Digital Humanities häufig genutzte Methode zur Analyse von Textkorpora, unter anderem zum Vergleich von Texten zwecks Zuordnung zu Epochen, ist die Stilometrie. Dabei wird die Ähnlichkeit der Texte mithilfe eines Distanzmaßes gemessen, wobei unter anderen das Burrows' Delta-mean Distanzmaß (Burrows 2002, S. 267) genutzt wird (weitere Distanzmaße sind das euklidische und das Cosinus Distanzmaß). Je kleiner die Distanz zwischen zwei Texten ist, desto ähnlicher sind sie sich. Mit der Distanz eines Textes zum Mittelwert des Korpus kann seine Epochenzuordnung ermittelt werden. Die Distanzmessung kann noch andere Erkenntnisse zutage fördern; u.a. soll man erkennen können, ob der Text von einer Frau oder einem Mann geschrieben wurde. Eine detailliertere Beschreibung der Methode und ein Anwendungsbeispiel finden sich auf dem Download Server.

3.7 Topic Modeling

> Topic Modeling ist ein auf Wahrscheinlichkeitsrechnung basierendes Verfahren zur Exploration größerer Textsammlungen. Das Verfahren erzeugt statistische Modelle (Topics) zur Abbildung häufiger gemeinsamer Vorkommnisse von Wörtern. Die Methode des Topic Modeling bietet die Möglichkeit, Textsammlungen thematisch zu explorieren. Dabei geht man davon aus, dass eine Textsammlung aus unterschiedlichen ›Themen‹ bzw. besser: ›Topics‹ besteht, die in den einzelnen Dokumenten der Sammlung in unterschiedlicher Ausprägung vertreten sind. Unter einem ›Topic‹ versteht man dabei eine Gruppe von Wörtern (wie zum Beispiel die Wörter »Theater«, »Schauspieler« und »Stück«), die in einem Text ungewöhnlich – d.h. statistisch auffällig – oft gemeinsam vorkommen.[22]

22 Horstmann 2018.

Zu dem Thema Topic Modeling oder Themenmodellierung existieren eine Vielzahl von Dokumentationen in wissenschaftlichen Werken (z.B. in Weitin 2021, S. 118ff.) oder auf der Seite *forText – Literatur digital erforschen*.[23]

Bei der Themenmodellierung werden zunehmend KI-Techniken eingesetzt, wodurch dieses Verfahren zum Gruppieren von Dokumenten in semantische Cluster häufig genutzt wird. Die Häufigkeitsverteilung von Wörtern in einem Textkorpus folgt dem Zipf'schen Gesetz, das vereinfacht besagt, dass wenige Wörter sehr häufig und viele Wörter eher selten vorkommen. Bei der Themenmodellierung sind nur Wörter im mittleren Häufigkeitsbereich von Interesse; die häufigsten Wörter sind in der Regel inhaltsleere Funktionswörter (Stoppwörter), und die seltensten Wörter sind so spezifisch, dass sie für das Modell nicht von Nutzen sind.

Anwendungsbeispiel

Es soll der Inhalt einer größeren Textsammlung – wie beispielsweise das Œuvre Therese Hubers oder auch die gesamte Prosaliteratur des 18. und 19. Jahrhunderts – erforscht werden. Das Topic Modeling ermöglicht es, die Texte untereinander zu vergleichen, ohne jeden Text der Sammlung individuell zu lesen. Abhängig von der Größe der Textsammlung kann man festlegen, wie viele ›Topics‹ erstellt werden und wie groß diese Topics sein sollen. Die Topics werden manuell so lange modelliert, bis ein aussagekräftiges Ergebnis erscheint (vgl. Weitin 2021). Nun kann untersucht werden, »welche Topics in welchen Texten besonders oft vertreten sind – oder auch umgekehrt, welche Texte ein gegebenes Topic besonders stark ›thematisieren‹« (ebd.); dabei können textimmanente Bedeutungsstrukturen in ihnen abgelesen werden.

Technische Grundlagen

Der im Topic Modeling am häufigsten genutzte Algorithmus wurde von Blei, Ng und Jordan (vgl. Blei 2003, S. 993ff.) unter dem Namen Latent Dirichlet Allocation (LDA) entwickelt und liegt auch den Tools Mallet machine learning for language toolkit)[24] (in Java programmiert) und DARIAH[25] (in Python programmiert) zugrunde. »Er basiert auf einer wiederholt zufälligen Auswahl an Textsegmenten, wobei innerhalb dieser Segmente jeweils die statistische Häufung von Wortgruppen erfasst wird. Der Algorithmus berechnet somit die Topics der Textsammlung, die Topic-Anteile in den Einzeltexten und welche Wörter zu den jeweiligen Topics gehören.« (Ebd.) In R wird das package topicmodels zur Verfügung gestellt, Microsoft setzt in seinem Azure Machine Learning Paket auch den LDA-Algorithmus ein.[26]

---

23 https://fortext.net/ [Zugriff: 30.10.2023]. Vgl. zu dem Thema auch den Artikel von Laura Kraft im vorliegenden Band.

24 https://mimno.github.io/Mallet/topics [Zugriff: 27.10.2022]

25 Vgl. DARIAH, TopicsExplorer – Themen und Inhalte von Textsammlungen erkunden, »In cooperation with DARIAH-DE«, URL: https://de.dariah.eu/text-analysis-with-topic-models [Zugriff: 27.10.2022]

26 Vgl. Microsoft (2022): »Latent Dirichlet Allocation«-Komponente, URL: https://learn.microsoft.com/de-de/azure/machine-learning/component-reference/latent-dirichlet-allocation [Zugriff: 27.10.2022]

4. Softwaretools und Entwicklungsumgebungen zur Textanalyse

Es werden in diesem Kapitel einige Softwaretools und Entwicklungsumgebungen für die Textanalyse vorgestellt und die wichtigsten Eigenschaften aufgelistet. Zu den Softwaretools gehören:

- CATMA (Verteilung, Nachbarn, Wordcloud, Annotationen im Team, u.v.a.m.)
- Stanford CoreNLP (Natural Language Processing)
- Python (Programmiersprache mit guter Unterstützung von Anwendungen zur Textanalyse) und spaCy (u.a. Tokenisierung, Lemmatisierung, POS (Part Of Speech) Tagging, Visualisierung)
- R Projekt und RStudio (Stilometrie, KI-Clustering)
- Visone (Netzwerkanalyse und -visualisierung)
- LitAs (Analyse von literarischen Texten und Dramen in Java)

Diese Liste erhebt keinen Anspruch auf Vollständigkeit, da es bereits eine unüberschaubare Anzahl von Softwaretools zur Analyse von (literarischen und nicht-literarischen) Texten auf dem Markt und im Bereich der *Digital Humanities* gibt. Die hier gelisteten Softwaretools werden nachfolgend teilweise nach den Rezensionsvorgaben von Homburg u.a. (2020) vorgestellt und rezensiert, indem die wichtigsten Eigenschaften der Softwaretools genannt und grob nach ihrem Nutzen für die Textanalyse bewertet werden. Eine vollständige Rezension und Evaluation der hier aufgeführten Tools kann an dieser Stelle nicht geleistet werden; dazu müssten alle Funktionen und Eigenschaften der Software systematisch von unterschiedlichen Nutzer:innen vom Key-User bis hin zum Standard-User getestet und bewertet werden (für eine vollständige Liste der Evaluierungskriterien, vgl. Homburg u.a. 2020). Die Rezension der Tools verfährt grundsätzlich nach einem festgelegten Schema mit den aufgelisteten Aspekten und folgt dabei im Wesentlichen den Rezensionsvorgaben von GitHub.[27]

- Aufgabe und Funktionen des Tools
- Ablauf der Nutzung des Tools: Download, Installation, Konfiguration, Import von Texten (Datenformate), Textanalyse, Export, Aufbereitung von Grafiken usw.; Möglichkeit von Ablaufprozessen
- Plattform: Betriebssystem, Datenbank, Web
- Anforderung an Hardware- und Softwareversionen
- Entwickler des Tools
- Programmiersprache des Tools
- Welche Methoden/Algorithmen wurden verwendet?
- Ggfs. im Tool nutzbare Programmiersprache(n)
- Open Source, Lizenzkosten, Kosten für Support und Wartung, Nutzermodell bei Lizensierung
- Schnittstellen (Import, Export, Datenbankzugriff, API usw.)

27 URL: https://research-squirrel-engineers.github.io/Impuls_SoftwareRezensionen_DGUF/Draft.html [Zugriff:10.08.2023]

- Konfigurierbarkeit, Einstellungen, Mehrsprachigkeit
- Bedienungsanleitung/Dokumentation
- Bedienbarkeit (u.a. wie werden Texte importiert)
- Nutzung von Grafiken
- Ggfs. Programm-Entwicklungsumgebung zur Nutzung des Tools
- Support und Hotline
- Datenschutz

4.1 CATMA

CATMA[28] (Computer Assisted Text Markup and Analysis) wird seit 2008 an der Universität Hamburg entwickelt[29] und ist ein frei verfügbares, webbasiertes Tool, mit dem man digitale Texte manuell annotieren, analysieren und visualisieren kann – allein oder kollaborativ im Team (vgl. Schumacher 2019).Eine gute Dokumentation ist in forText (vgl. Schumacher 2019) zu finden.

Die Annotationskategorien (sog. Tagsets) kann der Nutzer frei festlegen, so dass sich CATMA besonders für literaturwissenschaftliche Projekte eignet, in denen die Kategorien zum Untersuchungsgegenstand passend definiert werden (vgl. Jacke 2018).

In dem Projekt heureCLÉA[30] wurden automatische Annotationsfunktionen entwickelt und in CATMA integriert. Dazu wurde das »Machine Learning«-Verfahren genutzt, wobei drei der manuell annotierten Phänomenkomplexe (Tempus, Zeitausdrücke und zeitliche Ordnung) als Trainingskorpus dienten. »Das Projekt 3DH arbeitet an der Entwicklung komplexerer Visualisierungsoptionen, von denen eine Auswahl in CATMA integriert werden soll.«[31] In CATMA können auch Texte importiert werden, denen der/die Anwender:in Tags mittels NER (siehe oben) zugeordnet hat (vgl. Schumacher 2022).

Um einen bestimmen Literaturtext zu analysieren, wird in CATMA zunächst ein neues Projekt angelegt (»CREATE A NEW PROJECT«). Der zu analysierende Text wird dann im TEI-Format aus dem Textgrid-Repository[32] heruntergeladen und in das Projekt importiert. Nun können die CATMA-Funktionen auf den Text angewendet werden:[33]

- Definition von Analyse-Funktionen (»Analyze«) mit frei definierbaren Queries (Analyseabfragen); dabei Ermittlung der typischen statistischen Merkmale
- Nach individuell definierten Kategorien (z.B. anhand von verfügbaren oder selbst erstellten Tagsets), überlappende und widersprüchliche Annotationen sowie Mehr-

28 Vgl. CATMA, inzwischen Version 7, URL: https://catma.de [Zugriff: 06.02.2024]

29 Vgl. CATMA an der Universität Hamburg, URL: https://www.slm.uni-hamburg.de/germanistik/forschung/forschungsprojekte/catma.html [Zugriff: 27.10.2022]

30 Vgl. Projekt heureCLÉA (2013 – 2016), Uni HH und Heidelberg, Projektleitung: Prof. Dr. Jan Christoph Meister), URL: https://www.slm.uni-hamburg.de/germanistik/forschung/forschungsprojekte/heureclea.html [Zugriff: 25.10.2022]

31 URL: https://www.slm.uni-hamburg.de/germanistik/forschung/forschungsprojekte/catma.html [Zugriff: 05.11.2023]

32 Vgl. https://textgridrep.org/, z.B. https://textgridrep.org/browse/qmv7_0 für Kafka, Erstes Leid [Zugriff: 28.10.2023]

33 Vgl. URL: https://fortext.net/tools/tools/catma [Zugriff: 08.11.2023]

fachannotation einzelner Wörter und Passagen. Annotationen einzeln oder im Team (Funktion Annotate)
- Visualisierung mit Hilfe von KWIC (Keywords in Context, Nachbar-Wörter zu einer Wortliste), Wordcloud mit Wörterauswahl, DoubleTree
- Distribution (Verteilung der Wörter im Text in n (10) Segmenten bzgl. Anzahl Zeichen) mit Distributionsgraph

Kurzbeschreibung der Eigenschaften von CATMA:

- Aufgabe und Funktionen: siehe oben
- Ablauf der Nutzung des Tools: siehe oben
  Nutzung z.B. zu literaturwissenschaftlichen Forschungszwecken, an hermeneutische Verfahren angelehnt
- Plattform: Web
- Anforderung an Hardware und Softwareversionen: entfällt, da es webbasiert und frei verfügbar ist. Es kann mit allen gängigen Browsern genutzt werden, lediglich eine Anmeldung ist erforderlich
- Entwickler des Tools: University of Hamburg, Department of Language https://catma.de/webarchive/catma-4.0/home.html
- Programmiersprache des Tools: Java
- Methoden/Algorithmen: umfangreich[34]
- Open Source: ja, lizensiert mit GNU General Public License
- Lizenzkosten und Kosten für Support und Wartung: keine
- Schnittstellen: Import von Texten, Export von Analyseergebnissen
- Textformate: TEI, XML, TXT, PDF
- Konfigurierbarkeit: teilweise, z.B. bei definierbaren Queries oder Auswahllisten; Mehrsprachigkeit: nein, nur englisch
- Bedienbarkeit (u.a. wie werden Texte importiert): etwas gewöhnungsbedürftig, aber gute Tutorials in forText (vgl. Schumacher 2019). Die Sichtbarkeit der Funktionen ist eingeschränkt und das Fensterhandling nicht selbsterklärend. Zum Speichern der Annotationen muss das eye-icon der tagsets im *Annotate* module aktiviert werden, sonst keine Speicherung.
- Bedienungsanleitung: ja bei forText[35]
- Nutzung von Grafiken: ja viele
- Support: per Mail oder chat

Datenschutz: nach der EU-Datenschutz-Grundverordnung[36] In forText werden neben CATMA weitere Tools für die digitale Textarbeit aufgelistet und kurz beschrieben. Dort sind auch Links für den Download enthalten. Es handelt sich u.a. um Tools zur OCR-

34 URL: https://fortext.net/routinen/methoden [Zugriff: 15.04.2023]
35 URL: https://fortext.net/tools/tools/catma [Zugriff: 29.10.2023]
36 URL: https://digitallearninglab.de/tools/catma-computer-assisted-text-markup-and-analysis [Zugriff: 15.09.2023]

Schrifterkennung, Korpusanalyse, Netzwerkvisualisierung, Formatumwandlung, NLP-Funktionen, Stilometrie und Sentimentanalyse.

4.2 Stanford CoreNLP

Die Spracherkennung (NLP Natural Language Processing) ist für viele Bereiche ein wichtiger Bestandteil u.a. für Sprachübersetzungen oder für das Benutzerinterface in Geräten, Automobilen und Maschinen. Die elementare Erkennung von Wörtern, Wortarten, Sätzen usw. (Tokenisierung und Lemmatisierung) ist auch ein wesentlicher Bestandteil der Literaturtextanalyse.

Das Standford CoreNLP[37] ist ein Java-Tool zur Verarbeitung natürlicher Sprache. Es kann für Texte in mehreren Sprachen eine Tokenisierung, Lemmatisierung und Part-Of-Spech-Tagging durchführen, Named Entities erkennen (NER = Named Entities Recogintion), numerische und zeitliche Werte verarbeiten, sowie Dependenz- und Konstituenten-Parses, Koreferenz und Sentimentanalysen durchführen und auch Zitatattributionen und Relationen ermitteln. Der zu untersuchende Text kann dabei durch eine Pipeline mit hintereinander geschalteten Funktionen geschleust werden.

CoreNLP unterstützt derzeit acht Sprachen: Arabisch, Chinesisch, Englisch, Französisch, Deutsch, Ungarisch, Italienisch und Spanisch.

4.3 Python und spaCy

Python[38] ist eine moderne, objektorientierte, *high-level*, *general-purpose*, *Open-source*[39], Multiplattform Programmiersprache. Sie ist darüber hinaus auch eine mächtige Entwicklungsumgebung insbesondere in Verbindung mit dem Tool Anaconda für vielerlei Anwendungen u.a. aus dem Bereich Literaturtextanalyse. Dazu stehen eine Vielzahl von Ergänzungen in Form von *packages* und *libraries* (eine library enthält eine bestimmte Menge von packages zur Erfüllung einer Aufgabenstellung) zur Verfügung, z.B. spaCy für Tokenisierung, Lemmatisierung und POS-Tagging, neuerdings auch für KI-Anwendungen. Eine Web-Adresse für die Top 30 libraries ist in der Fußnote.[40] Die Python Software Foundation (PSF) ist eine unabhängige non-profit Organisation hinter Python, die die Copyrights auf Python innehat.[41]

SpaCy ist

> eine freie, quelloffene Bibliothek für fortgeschrittene natürliche Sprachverarbeitung (NLP) in Python. […]. Sie wurde speziell für den Einsatz in der Produktion konzipiert und hilft bei der Entwicklung von Anwendungen, die große Mengen an Text verarbeiten und »verstehen«. Es kann zur Erstellung von Systemen zur Informationsextraktion

37 Vgl. https://stanfordnlp.github.io/CoreNLP/ [Zugriff: 31.07.2023]

38 Vgl. URL: https://www.python.org/ [Zugriff: 30.10.2023]

39 Vgl. URL: https://opensource.org/ [Zugriff: 30.10.2023]

40 Vgl. URL: https://www.mygreatlearning.com/blog/open-source-python-libraries/ [Zugriff: 30.10.2023]

41 Vgl. URL: https://www.python.org/psf/ [Zugriff: 30.10.2023]

> oder zum Verstehen natürlicher Sprache oder zur Vorverarbeitung von Text für Deep Learning verwendet werden.[42]

Die Bibliothek ist leicht zu installieren, und ihre API (Application Programming Interface) ist einfach und produktiv. Sie eignet sich hervorragend für umfangreiche Informationsextraktionsaufgaben und wurde von Grund auf in Cython (CPython) geschrieben, das ein sehr gutes Speichermanagement enthalten soll. Cython ist eine universelle Prog rammiersprache, die weitgehend mit Python kompatibel ist. Wie diese unterstützt sie objektorientierte, aspektorientierte und funktionale Programmierung. Der Hauptvorteil liegt in der Übersetzung in die Zielsprache C, was das Programm deutlich performanter im Vergleich zum Standard-Python-Interpreter macht.[43]

Seit seiner Veröffentlichung hat sich spaCy zu einem Industriestandard entwickelt. Eine Vielzahl von Plugins können in Machine-Learning-Programme integriert werden, um damit eigene Komponenten und Workflows (automatisierte Abläufe) zu erstellen.spaCy bietet u.a. folgende Funktionen für die Literaturtextanalyse an, siehe https://spacy.io/usage/spacy-101:

- Linguistisch-motivierte Tokenisierung
- Part-of-Speech Tagging, Dependency Parsing, Lemmatisierung, Satzsegmentierung, Named Entity Recognition, Entity Linking, Wort-Vektoren und -Ähnlichkeiten, Textklassifikation, Rule-based Matching
- Morphologische Analyse
- Textzusammenfassung
- Eingebaute Visualisierer für Syntax und NER (Named Entity Recognition)
- Trainingsmodelle und Pipelines
- Unterstützung für mehr als 72 Sprachen

spaCy kann packageweise heruntergeladen werden, wobei die package-Funktionen in einer Python-Entwicklungsumgebung ausgeführt werden.

Als Beispiel sei hier das Modul Part Of Speech für deutsche Texte genannt. Folgender Beispielsatz »Python ist ein System mit vielen Paketen für die Textanalyse« wird in seine sprachlichen Bestandteile (Basiswörter) zerlegt. Mit der Python-Entwicklungsumgebung IDLE wird das Python-Programm (py-Datei) geöffnet. Vorher müssen noch

- das Package spacy mit pip install spacy und
- das deutsche Sprachpackage mit python -m spacy download de_core_news_sm

heruntergeladen und installiert werden. Das folgende kleine Python-Programm für NLP-POS wurde ebenfalls von der Webadresse von spaCy[44] heruntergeladen und lokal mit dem Beispieltext ausgeführt:

---

42 spaCy, Industrial-Strength Natural Language Processing, URL: https://spacy.io/ und https://spacy.io/usage/spacy-101 [Zugriff: 27.10.2022, Übers. v. JH]

43 Vgl. URL: https://dewiki.de/Lexikon/Cython [Zugriff: 05.11.2023]

44 Vgl. URL: https://spacy.io/ [Zugriff 30.10.2023]

```
import spacy
nlp = spacy.load("de_core_news_sm")
doc = nlp("Python ist ein System mit vielen Paketen für die Textanalyse")
for token in doc:
        print(token.text, token.lemma_, token.pos_, token.tag_, token.dep_,
                token.shape_, token.is_alpha_, token.is_stop)
```

Nach dem Start des Programms mit dem Kommando ›Run‹ lädt die Anweisung import spacy die zuvor heruntergeladene spaCy-library mit den NLP-Funktionen; spacy.load(…) lädt eine spezielle Bibliothek für deutsche Texte und weist sie der Variablen nlp zu. Mit der Anweisung doc = nlp(<Text>) werden die Tokens (Wörter) im <Text> (Python ist ein …) inkl. ihrer Eigenschaften (token.txt, token.lemma, …) ermittelt. Innerhalb einer for-Schleife, die über alle Wörter im <Text> geht, werden die ermittelten Werte wortweise angezeigt:

Tabelle 2: Ausgabe des o.g. Python-Programms für Part Of Speech

Text	Lemma	Pos	Tag	Dep	Shape	is_alpha	is_stop
Python	Python	NOUN	NN	Mo	Xxxxx	True	False
Ist	sein	AUX	VAFIN	ROOT	xxx	True	True
Ein	ein	DET	ART	Nk	xxx	True	True
System	System	NOUN	NN	Sb	Xxxxx	True	False
Mit	mit	ADP	APPR	Mnr	xxx	True	True
vielen	vieler	DET	PIAT	Nk	Xxxx	True	True
Paketen	Paket	NOUN	NN	Nk	Xxxxx	True	False
Für	für	ADP	APPR	Mnr	Xxx	True	True
Die	der	DET	ART	Nk	Xxx	True	True
Textanalyse	Textanalyse	NOUN	NN	Nk	Xxxxx	True	False

Quelle: Die Ausgabe wurde lokal mit dem o.g. POS-Python-Programm erstellt.

Die Spaltenüberschriften bedeuten[45]:

- Text: The original word text.
- Lemma: The base form of the word.
- POS: The simple UPOS part-of-speech tag (Wortarten).
- Tag: The detailed part-of-speech tag.
- Dep: Syntactic dependency, i.e. the relation between tokens.
- Shape: The word shape – capitalization, punctuation, digits.
- Is_alpha: Is the token an alpha character?
- Is_stop: Is the token part of a stop list, i.e. the most common words of the language.

45 URL: https://spacy.io/usage/linguistic-features und https://spacy.io/api/token#attributes [Zugriff 30.10.2023]

Statt eines direkten eingegebenen Textes kann auch eine Textdatei eingelesen und analysiert werden:

```
ner_text = open(<file path>). Read() # Datei öffnen und einlesen
doc = nlp(ner_text)
```

<file path> ist dabei der Pfad der Programmdatei. Obwohl spaCy für den Bereich NLP entwickelt wurde, können viele der Module für die Vorarbeiten zu Literaturtextanalyse genutzt werden.

Kurzbeschreibung der Eigenschaften von spaCy:

- Aufgabe und Funktionen des Tools, siehe oben
- Ablauf der Nutzung des Tools: Installation (pip install) der gewünschten Funktion und Nutzung in Python
- Plattform: Microsoft Windows©, macOS© und Linux
- Anforderung an Hardware und Softwareversionen: Installation von Python
- Entwickler des Tools: Explosion, siehe https://explosion.ai/
- Programmiersprache des Tools: Cython = C + Python
- Methoden/Algorithmen: siehe oben (hauptsächlich NLP)
- Im Tool nutzbare Programmiersprache: Python
- Open Source: ja
- Lizenzkosten: keine
- Schnittstellen (Import, Export, Datenbankzugriff, API usw., siehe u.a. https://spacy.io/api/). In einem Java-Programm kann ein Python-Programm mit Datenübergabe ausgeführt werden, z.B. um die spaCy-Funktionen in Python in einem Java-Programm zu nutzen.
- Konfigurierbarkeit, Mehrsprachigkeit: über Python oder API; 72 Sprachen
- Entwicklungsumgebung: Python mit u.a. IDLE
- Bedienbarkeit: via Pythoneditor
- Bedienungsanleitung: umfangreich im Web
- Nutzung von Grafiken: ja
- Support und Hotline: siehe https://github.com/explosion/spaCy
- Datenschutz: siehe https://github.com/explosion/spaCy/security/advisories

4.4 System R und RStudio

R (System R[46]) ist eine Programmiersprache und freie (GNU General Public License der Free Software Foundation) Software-Entwicklungsumgebung für statistische Berechnungen und Grafiken. R enthält eine Multiparadigmensprache der vierten Generation und »bietet eine Vielzahl von statistischen (lineare und nichtlineare Modellierung, klassische statistische Tests, Zeitreihenanalyse, Klassifizierung, Clustering, …) und

46 Vgl. URL: https://www.r-project.org/ [Zugriff: 09.08.2023]

grafischen Techniken an und ist sehr erweiterbar.«[47] In R können Abläufe/Prozesse mit Hilfe einer mächtigen Programmiersprache definiert und gestartet werden (R-Skripte). R läuft auf der UNIX-Plattform, auf Windows und MacOS.

RStudio ist eine integrierte Entwicklungsumgebung, quasi die GUI für die Statistiksoftware R. Es werden mehrere Fenster angezeigt, die beim Umgang mit R unterstützen und mehrere nützliche Werkzeuge enthalten. In R selbst steht eine Konsole mit der Möglichkeit, Skripte auszuführen, zur Verfügung. In RStudio werden standardmäßig vier Fenster angezeigt, je eins für R Skripte (oben links), für die angelegten Variablen und deren Werte (oben rechts), für eine Art Konsole für Kommando- und Skripteingaben (unten links) und eins für Informationen (unten rechts u.a. Help, packages, Plot-Ausgaben).

R kann um sogenannte *packages* erweitert werden; diese enthalten umfangreiche Bibliotheken für spezielle Aufgabenstellungen. Zum Beispiel kann für die Anwendung der Stilometrie in der Literaturtextanalyse das package stylo eingesetzt werden (vgl. Weitin 2021). Die Ergebnisse einer stilometrischen Analyse können grafisch mit Hilfe einer Clusteranalyse (Dendogramm) in R oder mit dem Tool Visone (siehe Kapitel 4.5) dargestellt werden:

> »Texte am gleichen Ast sind sich (im Sinne der Stilometrie) stilistisch ähnlich, je mehr Gabelungen zwischen zwei Texten liegen, desto unähnlicher sind sie sich«.[48]

Kurzbeschreibung der Eigenschaften von R und RStudio:

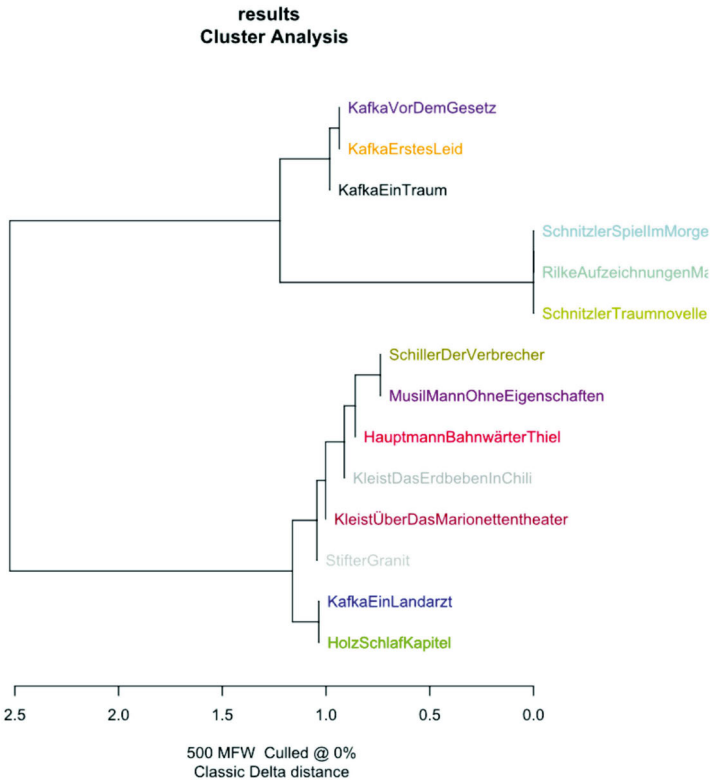
- Aufgabe und Funktionen des Tools: R ist eine freie Softwareumgebung für statistische Berechnungen und umfangreichen Visualisierungen (z.B. Grafiken); es ist quasi ein lokales System zur Nutzung von Bibliotheken für vielfältige Aufgabenstellungen
- Vorbereitung: Download, Installation, Ergänzung um packagies/libraries
- Nutzung des Tools mittels Eingabe von Kommandos oder Erstellung von ablauffähigen R-Skripts mit Hilfe der internen R-Sprache
- Plattformen: Microsoft Windows©, macOS© und Linux
- Entwickler des Tools: R Foundation, eine gemeinnützige Organisation, die im öffentlichen Interesse arbeitet
- Programmiersprache des Tools: R, Multiparadigmensprache der 4. Generation
- Entwicklungssprache: Java
- Open Source: ja
- Lizenzkosten und Kosten für Support und Wartung: keine
- Schnittstellen: Import und Export von Dateien u.a. Texten
- Konfigurierbarkeit: durch Parameter
- Bedienbarkeit: mittels umfassender GUI (RStudio)
- Nutzung von Grafiken: ja, Anzeige von sogen. Plots in einem der Fenster
- Support: FAQ und per Mail, Abonnement als supporting member

47 URL: https://www.r-project.org/about.html [Zugriff: 09.08.2023]

48 Horstmann 2019b, URL: https://fortext.net/routinen/lerneinheiten/stilometrie-mit-stylo

- Datenschutz: siehe https://support.posit.co/hc/en-us/articles/360042593974-R-and-R-Package-Security

Abb. 6: Clusteranalyse von 14 ausgewählten deutschsprachigen Texten mit den 500 häufigsten Wörtern, s. Anhang mit Stilometrie im Downloadbereich bei GitHub[49]; (Classic Delta Distance, 500 MFW, Culled 0 %)



Quelle: Grafik als plot mit System R stylo erstellt

4.5 Visone

Visone[50] wurde für die Analyse und Visualisierung von sozialen Netzwerken konzipiert und ist ein langfristiges Forschungsprojekt. Ein wesentliches Merkmal ist, dass Visone sowohl für Expert:innen als auch für Anfänger:innen die Möglichkeit schafft, ihre speziellen Netzwerkanalysen durch geeignete visuelle Methoden zu unterstützen.

Visone wurde in der Studie von Weitin (2021) und anderen das zur Darstellung der Ähnlichkeiten von Novellen, Romanen usw. auf der Basis von Distanzmaßen eingesetzt.

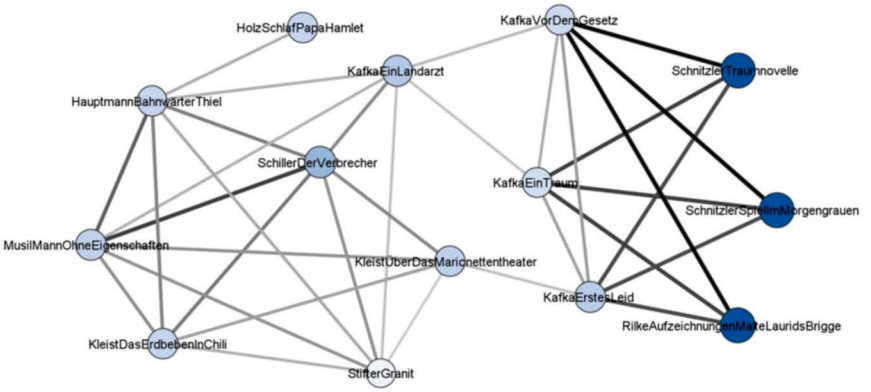
49 https://github.com/LuckeJohannes/Literaturwissenschaft_Informatik

50 Vgl. URL https://visone.ethz.ch/html/about.html [Zugriff: 09.08.2023]

Das verwendete Netzwerkmodell war ein Simmelian Backbone Network.
Hauptmerkmale der visone-Software sind:

- interaktive grafische Benutzeroberfläche, zugeschnitten auf soziale Netzwerke
- innovative Netzwerkvisualisierungen
- Unterstützung von unbestätigten Beziehungen
- verfügbar in Java für Windows, Linux und MacOS
- Import und Export von Standardformaten für soziale Netzwerkdaten
- Export in Publikationsqualität in JPEG, PDF, SVG, Metafile und andere Formate

Abb. 7: Visone-Darstellung des Simmelian Backbone Network von 14 ausgewählten deutschsprachigen Texten (s.o.) mit den 500 häufigsten Wörtern, s. Anhang mit Stilometrie im Downloadbereich bei GitHub[51]*; (Delta Distance, 500 MFW, Culled 0 %)*



Quelle: Grafik mit Visone auf Basis der mit R stylo erzeugten Distanzen erstellt.

Kurzbeschreibung der Eigenschaften von Visone:

- Aufgabe und Funktionen des Tools: Visone ist ein Tool, mit dem »Modelle und Algorithmen zur Integration und Weiterentwicklung der Analyse und Visualisierung von sozialen Netzwerken entwickelt werden«, vgl. https://visone.ethz.ch/html/about.html
- Download: das Tool kann heruntergeladen werden. Zum Ausführen ist eine Java Runtime erforderlich (jre).
- Plattformen: Microsoft Windows©, macOS© und Linux
- Entwickler des Tools: Zunächst war es ein Projekt der Universität Konstanz, inzwischen wird es von einem Netz von Kooperationen weiterentwickelt, an dem mehrere Universitäten beteiligt sind.
- Programmiersprache des Tools: Java
- Open Source: ja

51 https://github.com/LuckeJohannes/Literaturwissenschaft_Informatik

- Lizenzkosten und Kosten für Support und Wartung: keine
- Schnittstellen: Import von Dateien u.a. Adjazenzmatritzen als csv-Datei; Export von Grafiken und Dateien
- Konfigurierbarkeit: durch viele Einstellungen
- Bedienbarkeit: mittels umfassender Java-GUI
- Nutzung von Grafiken: Die Ausgaben sind meist Grafiken.
- Support: durch die DFG (Deutsche Forschungsgemeinschaft), BR 2158/1-1

4.6 LitAs

Das vom Verfasser selbst entwickelte Literaturtextanalysesystem LitAs ist zur Analyse von Literaturtexten und insbesondere von Dramen einsetzbar. Die Zielsetzung bei der Entwicklung des Programms war es neben der Einarbeitung in die Methoden der Literaturanalyse, ein Tool zu erstellen, das mehrere Analysefunktionen in einem Tool vereint. Es besteht aus einem in Java programmierten Analyseteil und einem Visualisierungsteil auf der Basis von Microsoft Excel©. Die Visualisierung umfasst u.a. ein Verteilungsdiagramm der Schlüsselwörter bzw. Dramenfiguren, eine Wordcloud sowie eine Konfigurations- und Adjazenzmatrix inkl. Grafiken bei Dramen. Im vorliegenden Artikel sind einige der Visualisierungen zu sehen (s.o).

Die Steuerung der Analyse wird mit einer Property-Datei geregelt, in der pro Steuerungselement eine Zeile mit vorgegebenen Werten enthalten ist, z.B.

```
Nachbarn=5
MinAehnlichkeit=70 (%)
Drama=ja
```

Die erste Zeile bedeutet, dass je 5 linke und rechte Nachbarn als Kontext ermittelt werden, die zweite gibt für die Ähnlichkeitssuche eine Prozentwahrscheinlichkeit für die Wortähnlichkeit vor, die dritte sagt aus, dass es sich um ein Drama handelt. Das Ergebnis der Analyse wird in eine txt-Datei ausgegeben, wobei die Daten für die Excel©-Visualisierung bereits passend zum Kopieren in die Exceldatei aufbereitet sind.
LitAs hat folgende Funktionen:

- Konfiguration mit Properties
- Import von Literaturtexten mit Tokenisierung, Lemmatisierung
- Export der Analyseergebnisse in txt-Datei
- Nutzung einer Stoppwortliste
- Ermittlung von Statistikwerten u.a. die (absolute) Häufigkeit der Wörter, absteigend sortiert
- Wortähnlichkeiten zu vorgebbaren Schlüsselwörtern mit Ähnlichkeits-Prozent
- Kontext der Schlüsselwörter mit n Nachbarn (analog zu CATMA KWIC) mit Häufigkeiten, absteigend sortiert
- Visualisierung mit Wordcloud, Verteilung der Schlüsselwörter, Anzahl Verteilungsbereiche vorgebbar (gleich lang oder gemäß der Szenenstruktur), Distributionsgraph

- Bei Dramen Verteilung der Figuren in den Szenen; welche Figur tritt mit welcher wie oft auf, Konfigurations- und Adjazenzmatrix inkl. Netzwerkgrafik

Kurzbeschreibung der Eigenschaften von LitAs:

- Aufgabe und Funktionen: siehe oben
- Plattform: Microsoft Windows©
- Entwickler: der Verfasser des vorliegenden Beitrags
- Programmiersprache: Java, Excel/VBA© für Visualisierung
- Lizenzkosten und Kosten für Support und Wartung: keine (Excel© vorausgesetzt)
- Schnittstellen: Import von Texten, Export von Analyseergebnissen
- Konfigurierbarkeit: durch Property-Datei
- Bedienbarkeit: mittels Pflege der Property-Datei und Import der Ergebnisse in Excel©-Templates
- Bedienungsanleitung: ja, kurz (erhältlich beim Verfasser)
- Nutzung von Grafiken: ja
- Support: ja (beim Verfasser)
- Datenschutz: nach Vereinbarung
- Die Lemmatisierung ist durch Nutzung der betr. spaCy-Funktion nlp() integriert (Zugriff aus Java auf Python-Programm)
- Methoden/Algorithmen: Tokenisierung, Konfigurations- und Adjazenzmatrix (s. Kap. 3.5), Wortähnlichkeit mit Jaccard-Koeffizient[52] und mit Levenshtein-Distanz[53]
- Open Source: ja (erhältlich beim Verfasser dieses Beitrags)

5. Plattformen, Projekte und Studien

Zur umfassenden Literaturtextanalyse wurden mehrere Projekte und Studien an Universitäten oder Forschungseinrichtungen definiert und gefördert.

Um Forschungsergebnisse zu speichern und auszutauschen, aber auch um Tools, Dokumentationen, Lehrinhalte usw. zur Verfügung zu stellen, haben sich eine Reihe von Plattformen etabliert, die für Forschung und Lehre genutzt werden

Zunächst werden die wichtigsten Plattformen im Bereich DH kurz vorgestellt, dann werden folgende Projekte und Studien beleuchtet:

- Projekt heureCLÉA (u.a. mit automatischen Annotationen anhand von jüngeren KI-basierten Technologien)
- Studie von Konle, Jannidis et al. (2021) (Disruptionen der Literaturwissenschaft am Beispiel der DVjs)
- Projekt GerDraCor (Netzwerkanalysen von deutschen Dramen, eingebunden in das europäische/internationale Projekt dracor.org mit einer Vielzahl von Analysewerkzeugen)

52 Vgl. URL: https://statologie.de/jaccard-koeffizient-python/ [Zugriff: 04.11.2023]

53 Vgl. URL: https://mathe.zone/data/ausarbeitungen/levenshtein-distanz.pdf [Zugriff: 04.11.2023]

- Projekt DARIAH (Schaffung einer digitalen Forschungsinfrastruktur für die Geistes- und Kulturwissenschaften)
- Projekt 3DH (Visualisierung und Exploration geisteswissenschaftlicher Daten in den Digital Humanities unter besonderer Berücksichtigung von 3D-Visualisierungsmethoden)
- Studie(n) von Weitin (2021, Digitale Literaturgeschichte. Eine Versuchsreihe mit sieben Experimenten)

5.1 Plattformen und Websites

Folgende Plattformen werden im Bereich DH für Forschung und Lehre genutzt; hier nur die bekanntesten:

- Forschungsdaten.info[54] ist ein deutschsprachiges Informationsportal zu Forschungsdatenmanagement (FDM). Diese Plattform wird von einer Vielzahl von deutschen Universitäten und Forschungseinrichtung unterstützt (Partner).
- Lehr- und Lernumgebungen für Digital Humanities[55] in forschungsdaten.info. Hier werden u.a. Links und Dokumentationen zu Programmiersprachen und -umgebungen (Python, System R, JupiterLab) angeboten.
- Base4nfdi: »Base4NFDI creates the basis for better of research data.« [56] »Die Nationale Forschungsdateninfrastruktur (NFDI) wird gemeinsam von Bund und Ländern finanziert und ist als bundesweit verteiltes Netzwerk organisiert. Ziel der NFDI ist der strategische Ausbau des Forschungsdatenmanagements in Deutschland, um Forschungsdaten zur erschließen und langfristig bereitzustellen.«[57]
- DH2go ist eine serverbasierte Lehr- und Lernumgebung an der Universität Stuttgart, um »Forschungssoftware, Trainingsdaten und Videos zentralisiert zur Verfügung zu stellen, wobei das gesamte System auch für die Nachnutzung bereitgestellt werden soll. DH2go eignet sich insbesondere zur Entwicklung in Python und R, zur Textverarbeitung (u.a. LaTeX, XML), zur Datenbankerstellung (MySQL, SQLite) und zur Visualisierung (Netzwerke mit Gephi, geografische Daten mit QGIS, diverse Darstellungen mit Python und R).«[58]

54 Vgl. URL: https://forschungsdaten.info/ [Zugriff: 05.11.2023]

55 Vgl. URL: https://forschungsdaten.info/wissenschaftsbereiche/geisteswissenschaften/tools-und-services/lehr-und-lernumgebungen-fuer-digital-humanities/ [Zugriff: 05.11.2023]

56 https://base4nfdi.de/

57 URL: https://forschungsdaten.info/fdm-im-deutschsprachigen-raum/deutschland/nfdi-nationale-forschungsdateninfrastruktur/ [Zugriff: 05.11.2023]. »Mit den Konsortien NFDI4Culture, NFDI4Objects, NFDI4Memory und Text+ sind vier geisteswissenschaftliche Konsortien an dem nationalen Vorhaben beteiligt.« (https://forschungsdaten.info/wissenschaftsbereiche/geisteswissenschaften/nfdi-konsortien/) [Zugriff: 03.03.2024]

58 URL: https://forschungsdaten.info/wissenschaftsbereiche/geisteswissenschaften/tools-und-services/lehr-und-lernumgebungen-fuer-digital-humanities/dh2go/ [Zugriff: 05.11.2023]

- DHVlab ist eine weitere Infrastruktur für Lehre und Forschung an der Universität München.[59]
- GitHub ist eine Plattform, auf der eigene Daten und Programme zur Verfügung gestellt werden können. Außerdem bietet sie die Möglichkeit zur Zusammenarbeit inkl. Versionskontrolle.[60]
- Mastodon ist ein Netzwerk, das sich als Alternative zu X (ehemals Twitter) etabliert hat. »Mastodon-Nutzer können kurze Texte, Bilder, Videos und andere Inhalte veröffentlichen, anderen Nutzern folgen und Beiträge kommentieren«[61]
- Zenodo ist ein vom CERN Data Center angebotenes kostenloses Open-source Repository, auf dem wissenschaftliche Daten, Software und Forschungsartefakte abgelegt und anderen zur Verfügung gestellt werden können. Dadurch sollen wissenschaftliche Prozesse in einem nicht-kommerziellen Umfeld unterstützt und beschleunigt werden.[62]
- Eine Alternative zu Zenodo ist der kommerzielle Datenspeicherdienst Figshare.[63]
- Zotero ist »ein kostenloses, benutzerfreundliches Tool, das Ihnen hilft, Forschungsergebnisse zu sammeln, zu organisieren, zu kommentieren, zu zitieren und zu teilen.«[64] U.a. ist auch eine Bibliotheksverwaltung zoterobib enthalten. Das Tool steht für viele Betriebssysteme zur Verfügung und kann heruntergeladen und installiert werden.[65]

5.2 Projekt GerDraCor[66]

Das Drama Corpus bzw. das German Drama Corpus (vgl. Fischer u.a. 2022) enthält eine Vielzahl von Dramen im TEI- oder txt-Format, wobei im deutschen Korpus ca. 500 Dramen gelistet sind. Die Texte können heruntergeladen werden, um sie lokal zu analysieren. Das Projekt DraCor enthält auch eine Reihe von Tools und grafische Anzeigen/Graphen, die für die Analyse der Dramen genutzt werden können. Detaillierte Informationen zum Projekt finden sich in forText (Horstmann 2020).

Zu erwähnen ist noch eine System R-Schnittstelle (API) zu DraCor[67], womit direkt aus R auf DraCor zugegriffen werden kann, um Texte oder Metadaten in geeignete R-Datenstrukturen herunterzuladen.

59 Vgl. URL: https://forschungsdaten.info/fdm-im-deutschsprachigen-raum/deutschland/bayern/serviceangebote/digital-humanities-virtual-laboratory-dhvlab/ [Zugriff: 05.11.2023]
60 Vgl. URL: https://github.com/ [Zugriff: 05.11.2023]
61 URL: https://www.tagesspiegel.de/gesellschaft/beliebte-alternative-zu-twitter-wie-funktioniert-das-netzwerk-mastodon-9059965.html [Zugriff: 04.11.2023]
62 Vgl. URL: https://zenodo.org/ [Zugriff: 05.11.2023]
63 Vgl. URL: https://figshare.com/ [Zugriff: 05.11.2023]
64 URL: https://www.zotero.org/ [Zugriff: 05.11.2023]
65 Vgl. URL: https://hochschulcloud.nrw/ [Zugriff: 05.11.2023]
66 URL: https://dracor.org/, https://dracor/ger.org/, [Zugriff: 31.10.2023]
67 URL: https://github.com/dracor-org/rdracor/ [Zugriff: 31.10.2023]

Grafische Anzeigen:

- Network: Netzwerk der gemeinsamen Auftritte; d.h. die Figuren eines Dramas kommen in der gleichen Szene vor, wobei die Dicke der Verbindung die Anzahl der gemeinsamen Auftritte repräsentiert.
- Relations: Visualisierung von Verwandtschafts- und anderen Beziehungsdaten
- Speech distribution: Verschiedene Möglichkeiten der Visualisierung der Sprachverteilung
- Full text: Voller Text im TEI-Format
- Downloads: Weitere spezielle Downloads

Tools:

- API (GET, PUT und Administrator-Funktionen)
- SPAROL (SQL-Schnittstelle)
- Easy Linavis: Simple Network Visualization for Literary Texts; erstellt einen Netzwerkgraphen zu gemeinsamen Auftritten von Figuren in Dramen und generiert daraus eine csv-Datei mit den Beziehungen
- Shiny Dracor mit der Auswahl eines Dramas und dann Anzeigen von
  - Graph (Gemeinsame Auftritte der Personen als Netzwerk-Graph mit Farben)
  - Edges (Gemeinsame Auftritte als Tabelle mit Zahlen)
  - Vertices (Tabelle mit Zwischenräumen)
  - Play Info: Ein weiteres Tool für die grafische Darstellung der gemeinsamen Auftritte[68]

5.3 Projekt heureCLÉA

HeureCLÉA war ein im Zeitraum 2013 bis 2016 vom BMBF gefördertes, Projekt, bei dem Literaturwissenschaftler:innen an der Universität Hamburg (Projektleiter: Jan Christoph Meister) und Informatiker:innen an der Universität Heidelberg (Projektleiter: Michael Gertz) interdisziplinär zusammen arbeiteten.[69] Es wurde das Ziel verfolgt, »die Möglichkeiten zu erforschen, die oft behauptete methodische Kluft zwischen qualitativer, hermeneutisch inspirierter Textanalyse in der Literaturwissenschaft und automatisierten, auf maschinellem Lernen basierenden Ansätzen in der Informatik, die textuelle Phänomene statistisch modellieren, zu überbrücken« (vgl. ebd.). »Diese Kluft existiert, wie wir beweisen konnten, nicht wirklich: Unsere beiden Herangehensweisen an Text schließen sich nicht gegenseitig aus, sondern stellen vielmehr komplementäre Positionen in einem methodologischen Kontinuum dar« (vgl. ebd.).
HeureCLÉA umfasste zwei Hauptarbeitspakete:

68 Für nahezu alle Dramen ist unter der URL https://weltliteratur.net/webweb/ger.html [Zugriff: 28.10.2022] ein konfigurierbares Simmelian Backbone Network zu finden, siehe Kapitel zu Visione.

69 Vgl. URL https://heureclea.de/index.html [Zugriff: 09.08.2023]

- (1) Korpuserstellung/kollaborative manuelle Annotation und
- (2) maschinelles Lernen/automatische Annotationen (vgl. ebd.).

Das Projekt heureCLÉA verbindet Fragestellungen aus der Narratologie der Literaturwissenschaft und der Sprachverabeitung (*Natural Language Processing*) aus der Informatik zur »Entwicklung eines heuristischen Moduls, das die Funktionalität der webbasierten Textanalyse und -annotationsumgebung CATMA 4 erweitert, indem es Vorschläge zu narratologischen Phänomenen in einem Text anbietet« und »neue[.] Data-Mining-Methoden für die bislang unbearbeitete Domäne literarischer Texte« (Gius/Jacke 2015) integriert.

In einem kollaborativen manuellen Annotationsansatz erstellten geschulte Annotator:innen ein annotiertes Korpus von beträchtlicher Komplexität. Es wurde das Annotationswerkzeug CATMA genutzt. Drei der sechs manuell annotierten narratologischen Kategorien wurden automatisiert, wobei das von Menschen erstellte Markup als Trainingskorpus für die Maschine diente. Drei dieser automatischen Annotationsfunktionen, wurden anschließend in CATMA integriert.[70]

5.4 Studie von Konle, Jannidis et al. (2021)

Konle, Jannidis u.a. veröffentlichten 2021 einen Artikel ›Disruptionen der Literaturwissenschaft am Beispiel der DVjs‹ in Melusina Press (vgl. Konle u.a. 2021).

Ein Ziel war dabei eine exemplarische Prüfung der Brauchbarkeit von Verfahren der digitalen Textanalyse in der Literaturwissenschaft (vgl. ebd., S. 1). Dazu wurde versucht, Disruptionen (Umbrüche, Strukturbrüche) in den Texten bzw. Beiträgen der DVjs von 1923 bis 2009 zu ermitteln (vgl. ebd., S. 6). Für die Erkennung von Disruptionen wurden Wort-Verteilungen (Häufigkeiten) mithilfe von Distanzmaßen der Stilometrie, z.B. der euklidischen Distanz, über einen historischen Entwicklungsprozess hinweg verwendet (vgl. ebd., S. 1). Dazu wurde ein Korpus gebildet, der die in der DVjs (Deutsche Vierteljahreszeitschrift für Literaturwissenschaft und Geistesgeschichte) veröffentlichten Artikel von 1923 bis 2009 beinhaltete. Daraus wurden Texte mit ca. 8000 Wörtern ausgewählt, von denen die ersten 3000 Wörter genommen wurden. Zusätzlich wurde das Tool Flair eingesetzt, um Eigennamen mit besserer Qualität zu erkennen. »Da wir Disruptionen inhaltlicher Natur suchen, werden die Wörter zusätzlich auf Substantive und Adjektive beschränkt, um stilistische Veränderungen möglichst abzuschwächen« (ebd., S. 7).

Es wurden nun die Häufigkeiten dieser Wörter und deren Veränderung über die Zeit untersucht. Dazu wurde folgendes Vorgehen gewählt:

- Bildung von 2 Textgruppen: 1. die ersten 10 Jahre 0–9, 2. die nächsten 10 Jahre 10–19
- Tokenisierung, Lemmatisierung, Satztrennung und POS-Tagging wurden mittels spaCy durchgeführt.

70 Vgl. URL: https://www.slm.uni-hamburg.de/germanistik/forschung/forschungsprojekte/heurecla.html [Zugriff: 31.10.2023]

- Nun wurde untersucht, wie stark sich die Texte des ersten Fensters von denen des zweiten unterscheiden; dazu wurde das euklidische Abstandsmaß (u.a.) mit den 5000/3000 häufigsten Wörtern genutzt.
- Annahme: Falls die Unterschiede sehr groß sind, dann liegt ein fachwissenschaftlich relevanter Umbruch vor.
- Anschließend wurden die beiden Zeitfenster um ein (später vier) Jahr(e) nach rechts verschoben und die Abstände erneut gemessen.

Die Texte wurden also »als Menge von Worthäufigkeiten repräsentiert […], [so dass] sich ein Text […] als Vektor in einem Raum mit ebenso vielen Dimensionen wie unterschiedlichen Wörtern« (ebd., S. 8) darstellen lässt (ebd., S. 8). Damit entspreche jedes Wort einer Vektor-Dimension, was die Verwendung von geometrischen Maßen wie euklidische oder Cosinus-Distanz möglich mache.

> Gleichzeitig stellt ein solcher Vektor, vorausgesetzt er enthält relative und keine absoluten Worthäufigkeiten, auch eine Verteilung über die Wahrscheinlichkeit für das zufällige Ziehen eines bestimmten Wortes dar (ebd., S. 8).

Da es keine valide Information darüber gibt, ob tatsächlich Disruptionen vorliegen, und somit das Ergebnis der quantitativen Analyse nicht bewertet werden kann, wurde zusätzlich eine Simulation vorgenommen. Dazu wurde ein künstliches Korpus erzeugt, der der Struktur der DVjs möglichst nahekommt, eine überschaubare Anzahl an Parametern hat und in den eine Disruption induziert wurde. Die künstlichen Texte wurden auf der Basis des Zipf'schen Gesetzes modelliert. Das Zipf'schen Gesetz besagt, dass wenn die Wörter eines Textes nach ihrer Häufigkeit geordnet sind, die Wahrscheinlichkeit p ihres Auftretens umgekehrt proportional zu ihrem Platz n (Rang) auf der Häufigkeitsliste ist: $p(n) \sim 1/n$.[71]

Mit diesem künstlichen Korpus wurde wieder das o.g. Vorgehen wiederholt, um herauszufinden, ob die induzierte Disruption gefunden wird. Dabei wurden verschiedene Distanzmaße (Euklidische und Cosinus Distanz) und auch andere Methoden wie Jensen-Shannon Divergenz, Pearson Korrelation und Machine Learning genutzt.[72] Um das Ergebnis noch zu optimieren, wurden zwei weitere komplexere Simulation mit Variation der künstlichen Texte durchgeführt.

Als Fazit wurde u.a. beobachtet, dass die Euklidische Distanz das beste Ergebnis bringt, wobei aber relativiert wurde, dass »die konzeptionelle Schlichtheit der Experimente eine künstliche Überbewertung der euklidischen Distanz erzeugt« (ebd., S. 15).

Die sich ergebende Methodik (Messung der Textdistanzen mit dem euklidischen Distanzmaß über versetzte Zeitfenster) wurde zum Schluss auf die DVjs angewendet.

Dabei wurde folgendes Ergebnis ermittelt:

---

71 Vgl. URL: https://bernardzitzer.com/de/zipfsche-gesetz-zipfs-law/ [Zugriff: 06.11.2023]

72 Vgl. zu Einsatz und ›Erfolg‹ diverser Distanzmaße, z.B. bei der Autorschaftsattribution, Jannidis u.a. (2021), S. 14 und Büttner u.a. (2017).

> Die Messung zeigt eine leicht erhöhte Distanz in den späten 1960er und den 1970er Jahren, die mit einem der in der Fachgeschichte beschriebenen Einschnitt, dem Szientifizierungsschub, zusammenfällt. Ansonsten ergibt sich hier […] ein erstaunlich homogenes Bild. Diese Beobachtung kann mehrere Gründe haben:
> 1. Die Veränderung der Aufsätze ist eher kontinuierlicher als disruptiver Natur.
> 2. Der Einfluss unbeobachteter Faktoren auf die Gestalt der Texte ist so groß, dass der untersuchte Parameter (Erscheinungsdatum) überlagert wird (ebd., S. 15).

Die Plausibilität der 2. Beobachtung wurde noch mit einem Topic Modeling verifiziert, das zwar keine Disruption, aber Trends und periodische Wiederholungen zeigte. Daraus wurde geschlossen, »dass die Verteilung von Worthäufigkeiten innerhalb eines Jahrgangs der DVjs multifaktoriell geprägt ist« (ebd, S. 17). Als Fazit wurde u.a. festgestellt, dass es »auch problematisch [sei], Aussagen zur Fachgeschichte aufgrund von nur einer Zeitschrift zu treffen« (ebd., S. 18).

Die Empfehlung ist, dass »künftige Simulationsstudien die Komplexität multifaktorieller Eigenschaften, wie sie die Topic Analyse der DVjs deutlich gemacht hat, besser abbilden« (ebd. S. 18) sollten u.a. den Aspekt der Autorschaft oder realistische Verteilungen bei den Texten.

Ferner wurde in diesem Zusammenhang die Unmöglichkeit einer systematischen Validierung durch annotierte Daten problematisiert und ein alternativer Weg der Auswertung durch Simulation vorgeschlagen.
In dem Projekt wurden dabei folgende Methoden und Tools genutzt:

Methoden:

- Topic Modeling
- Tokenisierung
- Lemmatisierung
- POS-Tagging (Part-of-speech-Tagging)
- Zipfsches Gesetz
- Distanzmessung von Texten

Neben den erwähnten Methoden wurden weitere Verfahren genutzt: Mahalanobis (Distanzmaß zwischen Punkten in einem mehrdimensionalen Vektorraum), Bhattacharya (Unterscheidung mehrerer mit demselben Wort bezeichneter Begriffe) und Kullback-Leibler Divergenz (bezeichnen ein Maß für die Unterschiedlichkeit zweier Wahrscheinlichkeitsverteilungen) sowie Manhattan Distanz. Die Manhattan Distanz wird verwendet, um die Unähnlichkeit zwischen zwei beliebigen Vektoren zu messen und wird häufig in Algorithmen für maschinelles Lernen eingesetzt (vgl. ebd., S. 8f).

Tools:

- Lyra (Schumacher/Becker 2021), ein Visualisierungstool zur Darstellung von Textdaten
- CATMA (s. Kap. 213)

- spaCy (vgl. Honnibal u.a. 2020) für Tokenisierung, Lemmatisierung, Satztrennung und POS-Tagging
- Support Vektor Maschinen (SVM)[73] für überwachtes maschinelles Lernen zur Textklassifikation

5.5 Projekt DARIAH[74]

DARIAH-DE (Digital Research Infrastructure for the Arts and Humanities) ist »eine Initiative zur Schaffung einer digitalen Forschungsinfrastruktur für die Geistes- und Kulturwissenschaften«.[75] Insbesondere werden Materialien für Lehre und Weiterbildung im Bereich der *Digital Humanities* entwickelt. Deshalb unterstützt DARIAH-DE die mit digitalen Methoden und Verfahren arbeitende Forschung in den Geistes- und Kulturwissenschaften mit einer Forschungsinfrastruktur, die i. W. aus vier Säulen besteht (vgl. ebd.):

- Lehre
- Forschung
- Forschungsdaten
- Technische Komponenten

Als Partner in DARIAH-EU trägt DARIAH-DE ferner dazu bei, europaweit *state-of-the-art* Aktivitäten der Digitalen Geisteswissenschaften zu bündeln und zu vernetzen. DARIAH wurde von 2011 bis 2021 vom BMBF gefördert (vgl. ebd.).
DARIAH bietet eine Vielzahl von Diensten und Werkzeugen an u.a.

- Online-Tool zur kollaborativen Texterstellung nach dem Motto: »Gemeinsam gleichzeitig an Texten arbeiten«
- Verwaltung von Mailinglisten für den Informationsaustausch innerhalb der Forschungsgemeinde
- Modellierung von Datenstrukturen: »Die Datenmodellierungsumgebung (DME) ist ein Werkzeug zur Modellierung und Verknüpfung von Daten. Mit Hilfe der DME werden Datenmodelle und Zuordnungen zwischen ihnen definiert und in Form von Schnittstellen (REST-API) bereitgestellt«[76]
- Helpdesk
- MYSQL-Datenbank-Hosting
- TopicsExplorer (Tool für das Topic Modeling)
- u.v.a.m.

---

73 Vgl.https://lmb.informatik.uni-freiburg.de/lectures/mustererkennung/WS0506/10_ME.pdf [Zugriff: 04.11.2023]

74 URL: https://de.dariah.eu/ [Zugriff: 01.11.2023]

75 Digital Research Infrastructure for the Arts and Humanities, »In cooperation with DARIAH-DE« URL: https://de.dariah.eu/ [Zugriff: 27.10.2022]

76 URL: https://marketplace.sshopencloud.eu/tool-or-service/YxrA2F [Zugriff: 01.11.2023, übers. v. JH]

DARIAH TopicsExplorer

Der DARIAH TopicsExplorer[77] ist ein Tool für die Themenmodellierung, das Themen (Topics) als Wahrscheinlichkeitsverteilungen über den gesamten Wortschatz eines Textkorpus ermittelt (LDA-basierte Textanalyse). Der TopicsExplorer hat dabei einen relativ einfachen Arbeitsablauf. Zur Nutzung kann man ihn (z.B. für Windows) herunterladen und mit eigenen Literaturtexten testen.

Im Folgenden wird ein Beispiel gezeigt, das inkl. Tool, Kurzdokumentation und Beispieldaten von GitHub heruntergeladen wurde.[78] Zur Ermittlung der Topics/Themen werden Trainingsdaten verwendet, die in einem KI-Prozess zum Lernen herangezogen werden. Es handelt sich um folgende 10 zufällig aus dem Novellenschatz ausgewählte Texte (siehe die Studie von Weitin 2021 in Kap.5.7):

- Brentano.txt: Geschichte vom braven Kasperl und dem schönen Annerl
- Droste.txt: Die Judenbuche
- Eichendorff.txt: Die Glücksritter
- Goethe.txt: Die neue Melusine
- Keller.txt: Romeo und Julia auf dem Dorfe
- Kleist.txt: Die Verlobung von St. Domingo
- Kurz.txt: Die beiden Tubus
- Schreyvogel.txt: Samuel Brinks letzte Liebesgeschichte
- Stifter.txt: Brigitta
- Storm.txt: Eine Malerarbeit

Es werden folgende Parameter vorgegeben:

- Anzahl Topics/Themen (hier 10)
- Anzahl der Iterationen des Lernprozesses (hier 100)
- Stoppwortliste mit 878 deutschen Stoppwörtern (wurde mit dem Download mitgeliefert)

Was noch fehlt, ist das Ausblenden der bekannten Namen, die das NER-Verfahren liefern würde. Das konkrete Ergebnis des TopicsExplorer sind zum einen die Topics/Themen, die nach der höchsten Wahrscheinlichkeit sortiert angezeigt werden (siehe Abb. 8) (wobei ein Topic ja aus einer Kombination von Wörtern besteht).

Man kann nun per Klick auf ein Topic durch Themen und Dokumente navigieren, sich ähnliche Themen und Dokumente anzeigen lassen und Auszüge aus den Originaltexten lesen. Wenn man z.B. auf eines der Topics klickt, sieht man »die 15 relevantesten Wörter für dieses Thema, sowie die 10 relevantesten Dokumente, deren Balkenbreite die jeweilige Gewichtung angibt, und die drei ähnlichsten Themen, bei denen die Kosinusähnlichkeit zwischen allen Themenvektoren berechnet und gewichtet wurde« (Anzeige des Tools bei dieser Aktion).

77 Vgl. DARIAH TopicsExplorer – Themen und Inhalte von Textsammlungen erkunden, URL: https://de.dariah.eu/text-analysis-with-topic-models [Zugriff: 27.10.2022]

78 URL: https://dariah-de.github.io/TopicsExplorer/ [Zugriff: 01.11.2023]

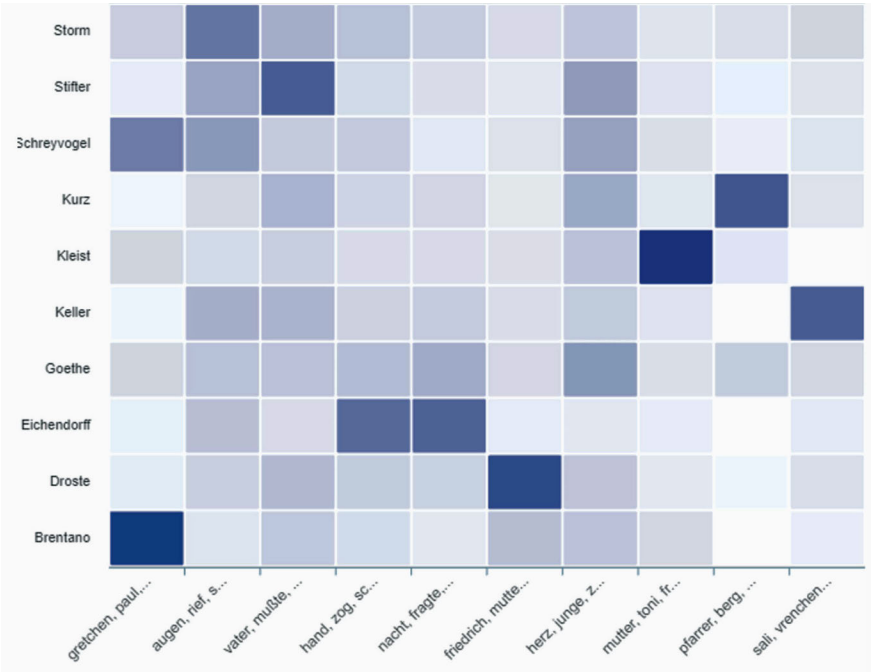
Man kann überdies die folgende Dokument-Themen-Verteilung (Document-Topic Distribution) betrachten[79]:

Abb. 8: 10 Topics in 10 Dokumenten

Quelle: Aus den Ergebnisdaten vom Tool generierte Grafik

79 URL: https://dariah-de.github.io/TopicsExplorer/ [Zugriff: 01.11.2023]

Abb. 9: Dokument-Themen-Verteilung[80]



Quelle: vom Tool generierte Grafik

> Jedes Dokument besteht bis zu einem gewissen Grad aus jedem Thema, was eine der theoretischen Annahmen von Themenmodellen ist« (Anzeige im Tool). Kleine Werte sind aus Gründen der Visualisierung auf null gerundet. »Die Visualisierung der Dokument-Themen-Verhältnisse in einer Heatmap zeigt die Art von Information, die […] über die reine Exploration hinaus[geht] und […] verwendet werden [kann], um thematische Entwicklungen über eine Reihe von Texten aufzuzeigen, ähnlich wie ein dynamisches Themenmodell (Anzeige im Tool).[81]

Kurzbeschreibung der Eigenschaften von DARIAH TopicsExplorer (nach dem Rezensionsschema in Kap. 3):

- Aufgaben und Funktionen des Tools (s.o.)
- Plattformen: Microsoft Windows©, macOS© und Linux
- Entwickler des Tools: DARIAH.de wird von 16 Instituten getragen, die von der SUB Göttingen koordiniert werden[82]
- Programmiersprache: Python
- Datenbank: SQLite

80 URL: https://dariah-de.github.io/TopicsExplorer/ [Zugriff: 01.11.2023]
81 URL: https://dariah-de.github.io/TopicsExplorer/ [Zugriff: 01.11.2023]
82 Vgl. URL https://de.dariah.eu/impressum [Zugriff: 09.08.2023]

- Open Source: ja
- Lizenzkosten und Kosten für Support und Wartung: keine
- Konfigurierbarkeit: ja
- Schnittstellen: Import von Texten, Exportfunktion
- Bedienbarkeit: mittels einfacher GUI
- Nutzung von Grafiken: ja
- Support: fachlicher und technischer Support per Mail, siehe https://de.dariah.eu/support
- Datenschutz: siehe https://de.dariah.eu/datenschutz

5.6 Projekt 3DH

3DH, Three-Dimensional Dynamic Data Visualisation and Exploration for Digital Humanities Research[83] war ein Projekt der Universität Hamburg von 2016 bis 2018. »Forschungsgegenstand ist die dynamische Visualisierung und Exploration geisteswissenschaftlicher Daten in den Digital Humanities unter besonderer Berücksichtigung von 3D-Visualisierungsmethoden« (Meister 2016).

Mit 3DH sollte eine »theoretisch-methodische Orientierung« (Meister 2016) für Visualisierungen geschaffen werden, die auf die konzeptionellen Bedürfnisse der Geisteswissenschaftler:innen abgestimmt sind. Dazu sollten Prototypen von Visualisierungswerkzeugen entwickelt werden, die als Demonstrationsobjekte dienen können. In den letzten Jahren hat der Einsatz digitaler Methoden im Rahmen von *Digital Humanities*-Projekten immer mehr Daten produziert, so dass die *Big data*-Technik für die Geisteswissenschaften zunehmend relevant wird (vgl. Meister 2016).

Die Kernaussage dort ist, dass es zwar schon viele Visualisierungstools gibt (Word Clouds, Diagramme zur Statistik, u.v.a.m.), die beim ersten Betrachten einen gewissen Wow-Effekt erzeugen, »methodisch wie epistemologisch jedoch letztlich opak bleib[en].« (Meister 2016)

Die im 3DH-Projekt gelisteten Software-Tools werden über die Seite des Projektleiters Jan Christoph Meister angezeigt (vgl. ebd.), beschrieben und stehen auch zum Download bereit. Es handelt sich um folgende Tools

- Tempusmarker »für die Annotierung (»Tagging«) von Temporaloperatoren und anderen auf der Textoberfläche nachweisbaren Textbestandteilen, die wir als Konstituenten des sog. ›Temporalitätseffekts definieren‹[84]
- TempusParser und Plotter mit flash-demo; »ein neuer Visualisierungsansatz, um die Eigenart der Verarbeitung textuell repräsentierter Zeitinformation darzustellen, wie sie insbesondere in Ereigniserzählungen gegeben sind«[85]
- EpiTest & EventParser[86]

83 Vgl. URL: https://threedh.net/ [Zugriff: 28.10.2022]
84 URL: https://jcmeister.de/tempusmarker/ [Zugriff: 01.11.2023]
85 URL: https://jcmeister.de/tempusparser-plotter/ [Zugriff: 01.11.2023]
86 Vgl. URL: https://jcmeister.de/epitest-eventparser/ [Zugriff: 01.11.2023]

Das Programm EpiTest wurde entwickelt, um eine kombinatorische Analyse von *.esf-Dateien, die von EventParser erzeugt wurden, durchzuführen. Der Kern des Programms besteht aus mehreren Algorithmen, die gespeicherte Datenbanken von Verbindungs-EVENTS suchen, die den Kategorien in der Episodenmatrix entsprechen (vgl. ebd.)

5.7 Studie(n) von Weitin (2021)

Im Jahr 2021 veröffentlichte Thomas Weitin eine Reihe von Experimenten der Literaturtextanalyse mit digitalen Methoden mit dem Titel *Digitale Literaturgeschichte, eine Versuchsreihe mit sieben Experimenten* (vgl. Weitin 2021). Darin beschreibt er die Ziele, das Vorgehen und die Ergebnisse der anhand von digitalen Methoden und Tools durchgeführten Experimente.

Im ersten Experiment sollte ermittelt werden, wie groß die Unterschiede zwischen je zwei Novellen aus einem Korpus sind, das aus 86 Novellen des *Deutschen Novellenschatzes* besteht (ab 1871, Herausgeber Paul Heyse und Hermann Kurz). Eine der Novellen ist *Die neue Melusine* (geschrieben 1807/1808) von Goethe. Es soll speziell ermittelt werden, ob diese Novelle »stilbildend für den Rest der Novellensammlung« ist und damit als »zentrale Referenz für die Integration romantischer Novellen verwendet« (Weitin 2021, S. 22) werden kann.

Um diese Frage zu beantworten, wird ein *Simmelian Backbone-Netzwerk* des Deutschen Novellenschatzes (Modell 1) verwendet, bei dem die Knoten die Novellen sind und die verbindenden Kanten zwischen je zwei Novellen durch Farbgebung anzeigen, wie ähnlich sich diese beiden Novellen bezüglich des (stilometrischen) Stiltyps, gemessen mit einem Distanzmaß, sind (dunkle Verbindung = große, helle = wenig Ähnlichkeit). An den Knoten liegt zusätzlich der Wert für die Ähnlichkeit jedes einzelnen Textes zum Gesamtkorpus an (dunkler Knoten = hohe Korpusähnlichkeit, heller Knoten = diese Novelle hebt sich vom Mainstream der Sammlung ab). Das Netzwerk wird auf Basis der Distanzen mit Hilfe des Netzwerkanalysetools ›Visone‹[87] erstellt (vgl. ebd.).

Die Ähnlichkeit wird mit Hilfe des Distanzmaßes Delta-mean, ein Abstandsmaß nach Burrows (vgl. Burrows 2002, S. 267ff.) ermittelt. Dabei werden aus den relativen Häufigkeiten der 500 häufigsten Wörter (MFW = Most Frequent Words) im Novellenschatz der Abstand einer Novelle zum Durchschnitt des Korpus sowie die Abstände von je zwei Novellen zueinander berechnet. Mit einem sogen. Culling-Wert von 20 % wird sichergestellt, dass ein MFW-Wort nur dann verwendet wird, wenn das Wort in mindestens 80 % aller Novellen vorkommt. Bei den Worthäufigkeiten werden zunächst alle Wörter, auch Artikel, Pronomen, Konjunktionen, Präpositionen und die Formen von sein und haben genutzt.

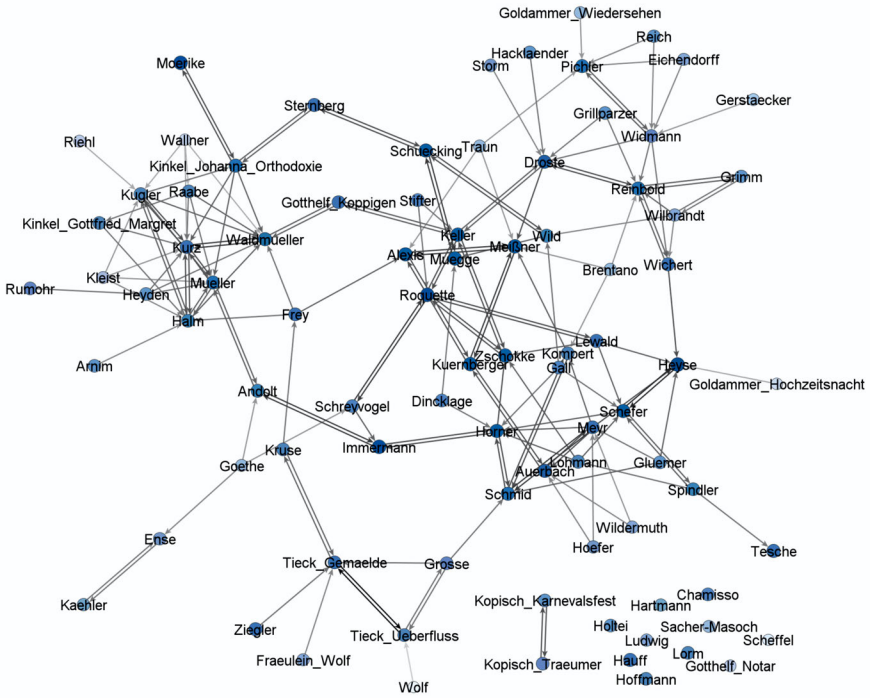
Zur Analyse der Novellen nach den genannten Verfahren wurde das System R zusammen mit der GUI RStudio (siehe Kap. 4.4) eingesetzt. R wird dabei um ein spezielles package namens »stylo« (Stylometric Multivariate Analyses) ergänzt, das von der R-Web-

87 Vgl. URL www.visone.de [Zugriff: 09.08.2023]

seite CRAN zur Verfügung gestellt wird und die nötigen Funktionen für die genannte Aufgabenstellung umfasst. Die Webadressen dazu sind in der Fußnote zu finden.[88]

Es wird eine Webadresse[89] genannt, von der sämtliche im Projekt verwendeten Daten wie die Korpora, die R-Skripte, mit denen die Analysen durchgeführt wurden, die Konfigurationen usw. sowie die Auswertungen der Analysen (Netzwerke, Plots u.a.) heruntergeladen werden können, um die Ergebnisse bei Bedarf nachzuvollziehen (vgl. ebd., S. 10), was mit entsprechender Einarbeitung in die verwendeten Tools dem Verfasser dieses Artikels auch gelang.

Abb.10: Simmelian Backbone Netzwerk nach der Simmel-Transformation



Quelle: Modifizierte Grafik nach Weitin 2021, S. 28.

In diesem Artikel wird nicht näher auf die Interpretation der Ergebnisse eingegangen, das ist die Aufgabe von Literaturwissenschaftler:innen. Aus Sicht der Informatik erstaunlich werden ausschließlich aus stilometrischen Ähnlichkeiten, die mit bestimmten Distanzmaßen mit Hilfe der Worthäufigkeiten ermittelt werden, viele interessante Erkenntnisse abgeleitet, wobei doch nur ein Parameter (Worthäufigkeit) genutzt wird. Unter anderem werden Autor:innengruppen und ihre Zugehörigkeit zur Romantik von der

88 Vgl. die URLs https://www.r-project.org/, https://rstudioapp.com/, https://cran.r-project.org [Zugriff 09.08.2023]

89 URL: https://github.com/thomasweitin/Digitale_Literaturgeschichte [Zugriff: 01.11.2023]

Analyse erkannt. Dazu wird ein Filteralgorithmus nach Georg Simmel genutzt, mit dem Netzwerke so transformiert werden, dass Gruppen sichtbar werden (vgl. Weitin 2021, S. 27). Aus dem vollständigen Netzwerk werden dabei nur die Verbindungen herausgefiltert, die »zu lokalen Ähnlichkeitsgruppen gehören« (ebd.), wobei ein spezieller Algorithmus nur Verbindungen innerhalb von Dreiecken sucht.

> Wir sehen zum Beispiel in unserem Modell in Abb. 2.1 [hier Abb. 10], dass Reinbold (2 Uhr) in ihrer Gruppe Teil von insgesamt 5 Dreiecken ist, während Goethe (8 Uhr) zu keinem einzigen Dreieck gehört. Heyse (3 Uhr) hat auch nur zwei Dreiecke, während Kurz (10 Uhr) in seiner ausgesprochen dicht vernetzten Gruppe Teil von 19 Dreiecken ist, die der Betrachter gar nicht so leicht auseinanderhalten kann. (Weitin 2021, S. 29)

Als Ergebnis der Analysen zum ersten Experiment wurde u.a. folgendes zusammengefasst:

- Heyses Novelle ist der durchschnittlichste Text des *Novellenschatzes*, weil er den geringsten Abstand (Delta mean) zum Durchschnitt des Korpus hat.
- Allerdings hat die Novelle von Kurz, die dem Korpusdurchschnitt wesentlich weniger entspricht als die von Heyse, deutlich mehr ›Follower‹ in der betreffenden lokalen Ähnlichkeitsgruppe.
- Wider Erwarten ist die Novelle ›Die neue Melusine‹ von Goethe nicht zentral im Netzwerk angeordnet und damit – so wird gefolgert – nicht stilbildend für die Novellensammlung (vgl. Weitin 2021, S. 24ff.).

Weitin weist darauf hin, dass das ausgewählte Korpus eher nicht repräsentativ sei: »Es ist gerade nicht die Repräsentativität, sondern die Exemplarität dieser Sammlung, die hier aufschlussreich ist, wenn man sie parallel zur Lektüre nach quantitativen Methoden modelliert« (Weitin 2021, S. 36).

Im zweiten Experiment »Autorennetzwerke im 18. Jahrhundert und in der Goethezeit« wurden Goethes Romane im Kontext ihrer Zeit, also in den Jahren 1770–1830, und ein Korpus mit Romanen aus dem 18. Jahrhundert untersucht (vgl. ebd., S. 40). Die Analysen wurden wieder mit dem System R durchgeführt. Für die Korpora wurde auf vorhandene digitale Ressourcen wie Textgrid und Gutenberg.de zurückgegriffen, an einigen Stellen konnte der automatische Schrifterkennungs-Workflow im Darmstädter LitLab genutzt werden. Bei dem Experiment ging es u.a. um die Fragestellung, »wie sich die Texte aus ein und derselben Feder in einem Korpus mit vielen verschiedenen Autorinnen und Autoren [bzgl. der Distanzmaße] zu einem Cluster zusammenfinden« (Weitin 2021, S. 38).

Die Ähnlichkeiten der 86 Texte aus einem Romankorpus zum 18. Jahrhundert wurden wieder in einem Simmelian Backbone-Netzwerk wie im 1. Experiment (s. oben) dargestellt. Dabei traten überraschende Erkenntnisse zutage: Es haben sich visuell bestimmte Gruppen herausgebildet, z.B. eine Gruppe sämtlicher im Korpus vertretenen Autor:innen (»Fund«, »Autorinnen-Netzwerk«). Allerdings fiel auf, »dass die Unterschiede im Wortgebrauch teilweise gar nicht inhaltlicher Natur waren, sondern durch unterschiedliche Schreibweisen und Modernisierungsstufen verursacht wurden« (Wei-

tin 2021, S. 45). Speziell die Schreibweisen mit/ohne th, ei/ey, Mut/Muth, Träne/Thräne u.a. sind hier zu nennen; dabei wurde versucht, die Effekte durch Änderung der Korpus-Zusammenstellung und Analyseparameter zu eliminieren.

Im weiteren Text von Weitins Buch wurden folgende Themen behandelt, auf die hier nicht näher eingegangen wird:

- Stilometrie mit Parametermanipulation und zwei Distanzmaßen im Vergleich
- Textklassifikation durch maschinelles, überwachtes Lernen mit automatischer Klassifikation nach der Variable Gender
- Quantitative Semantik mit Topic Modeling als Reading at Scale
- Semantische Netzwerkmodelle

Ein wichtiges Fazit von Weitin ist, dass das ›black box-Verhalten‹ der Analysetools kritisch sei, weil der Analyst »weniger gut in der Lage [sei] zu verstehen, wie Ergebnisse zustande kommen, sodass es auch schwieriger wird, Resultate mit kritischer Kompetenz zu interpretieren« (Weitin 2021, S. 39). Auch die wissenschaftliche Reproduzierbarkeit sei nicht immer gegeben.

6 Neue Anforderungen an Softwareprogramme zur Textanalyse

Neben den hier beschriebenen IT-Lösungen für die Analyse von Literaturtexten existiert eine Vielzahl weiterer Tools – vermehrt auch KI-basiert – oder solchen, die sich noch in der Entwicklung befinden. Es sind immer noch viele literatur-, geistes- und kulturwissenschaftliche Fragestellungen offen, für die es keine oder nur unzureichende Lösungen gibt – wie das Thema Bedeutung und Interpretation von literarischen Texten, die Genderfrage bei Texten unbekannter Autor:innen, die genaue historische Einordnung und Datierung von literarischen Texten unbekannter Autor:innen oder das Ermitteln von relevanten historischen Kontexten und ihrer Bedeutung für einen literarischen Text. Beispiele für Analyseanforderungen könnten das Erkennen von spezifischen Bedeutungen, rhetorischen Mitteln oder ästhetischen Strategien in Texten und das Erkennen von Widersprüchen sein. Auch komplexe literatur- und kulturwissenschaftliche Fragestellungen, die eine Analyse von narratologischen, semantischen, ästhetischen und kulturellen Phänomenen, Symboliken und Artefakten in Texten nach sich ziehen, stellen eine Herausforderung für die Informatik dar.

Neue Anforderungen an Softwaretools müssen spezifisch formuliert und in einem gewissen Maße formalisiert werden, damit sie klar und eindeutig sind. Um das zu gewährleisten, kann eine Anforderungsanalyse durchgeführt werden, bei der spezielle NLP-basierte Tools genutzt werden können (vgl. Femmer 2017). Auf Basis solcher spezifischen Anforderungen erstellen Informatiker:innen geeignete Softwaretools. Sie benötigen deswegen klare und eindeutige Anforderungen, da sie i.d.R. kein Fach- und Kontextwissen haben, d.h. keine (vertieften) Kenntnisse von literaturwissenschaftlichen Frage- und Problemstellungen, Themen und Gegenständen. Auch zum Testen des fertigen Tools werden die Anforderungen benötigt, ›gegen die praktisch getestet wird‹, damit sie sich in der Praxis bewähren können.

7 Fazit

Der Einsatz von IT-Programmen zur Analyse von literarischen Texten bietet zweifellos neue erkenntnis-, aber auch voraussetzungsreiche Möglichkeiten für die Literaturwissenschaft. Dabei sollte für eine Analyse genau festgelegt werden, welche Fragen (Anforderungen) beantwortet werden sollen. Das anzuwenden, was die IT bereitstellt und ermöglicht, ohne die dazugehörige Fragestellung in Anforderungen zu spezifizieren, bringt zwar Ergebnisse, deren Nutzungsgrad aber begrenzt und teilweise fragwürdig erscheint. Ein wichtiger Aspekt der angewendeten Methodik ist auch, dass die erzielten Analyseergebnisse valide sind. Um das zu prüfen, sollte die Analyse transparent dokumentiert werden (inkl. Daten und Tools), damit die Ergebnisse reproduziert werden können.

Eine Prüfung der eingesetzten Methoden auf Validität wird schwierig, wenn keine IST-Werte vorhanden sind wie in der Studie von Konle, Jannidis u.a.: Es war noch nicht literaturwissenschaftlich belegt, ob in dem verwendeten Korpus überhaupt Disruptionen existieren. Wenn in der Analyse festgestellt wird, dass es Disruptionen gibt, dann kann aber nicht sicher überprüft werden, ob das auch tatsächlich stimmt.

Es gibt inzwischen wie beschrieben viele Tools, oft mit speziellen Erweiterungen (CATMA, System R mit stylo, Python, spaCy), aber die Nutzung kann für Informatik-Laien eine außerordentliche Herausforderung werden. Im Fall von KI-basierten Tools gibt es bereits trainierte Modelle/Netze (z.B. für Annotationen), die genutzt werden können. Ein Training für überwachtes Lernen durchzuführen, ist dagegen schwierig und aufwendig. Hier kann eine Zusammenarbeit mit Wissenschaftler:innen aus beiden Disziplinen überaus nützlich sein.

Auf der Internetseite forText wird das Thema Stilometrie und seine Anwendung in der Literaturtextanalyse ausführlich erklärt und diskutiert. Es wird die Frage gestellt, ob »Stylo für DH-Einsteiger*innen geeignet« (ebd.) sei. In einer Checkliste wird bewertet, ob eine »Intuitive Bedienbarkeit« vorliegt (Ergebnis: »teilweise«) oder ob der Einstieg leicht ist (»nein«). Es wird darauf hingewiesen, dass »[g]rundlegende Kenntnisse in der Programmiersprache R […] zum Installieren und Starten des Tools vonnöten« sind (Horstmann 2019c). Es wird auch diskutiert, wie »etabliert Stylo für die digitale Stilometrie in den Literaturwissenschaften sei« (ebd.). Obwohl Stylo »eines der etabliertesten Tools« ist, findet es »wie die meisten digitalen Textanalysetools […] keine Erwähnung in Publikationen von Zeitschriften der traditionelleren Literaturwissenschaft« (Horstmann 2019c, S. 2). Im Artikel von Laura Kraft im vorliegenden Band wird auch auf die notwendige Code-Kompetenz hingewiesen und darauf, »dass einschlägiges Wissen über die jeweilige digitale Methode sowie den ihr zugrunde liegenden Algorithmus mindestens hilfreich, aber auch nahezu unerlässlich ist.«

Nachfolgend sollen hier noch einige Aussagen aus den weiter oben beschriebenen Projekten aufgeführt werden, die die Problematiken (z.B. der Validität) bei der kombinierten Anwendung von qualitativen und quantitativen Methodiken beider Fachdisziplinen auf den Punkt bringen (vgl. Konle, Jannidis et al. 2021):

- Die euklidische Distanz bringt gute Ergebnisse, aber: »die konzeptionelle Schlichtheit der Experimente kann eine künstliche Überbewertung der euklidischen Distanz erzeugen.« (Jannidis, F. u.a. 2021).
- »Es ist problematisch, Aussagen zur Fachgeschichte aufgrund von nur einer Zeitschrift zu treffen« (ebd.). Hier geht es um ausreichendes und repräsentatives Datenmaterial.
- »Künftige Simulationsstudien [müssen] die Komplexität multifaktorieller Eigenschaften, wie sie die Topic Analyse der DVjs deutlich gemacht hat, besser abbilden (ebd., S. 18). Bei Simulationen mit künstlichen Texten sollte auch auf eine »realistische Verteilung« (ebd., S. 18) bei den Texten geachtet werden; eine Normalverteilung liegt i.d.R. nicht vor (vgl. Weitin 2021, S. 67).
- Das Distanzmaß Kosinus-Delta ist sehr erfolgreich bei der stilometrischen Autorschaftsattribution, d.h. der Zuordnung von Texten zu (deren unbekannten) Autor:innen: »Der unter dem Pseudonym Robert Galbraith erschienene Kriminalroman *The Cuckoo's Calling* konnte via Stilometrie Joanne K. Rowling zugeordnet werden; Rowling hat sich daraufhin zu ihrer Autorschaft bekannt.« (Büttner u.a. 2017).

Digitale Tools für die quantitative Textanalyse einzusetzen ist wie schon ausgeführt eine methodische Herausforderung für Literaturwissenschaftler:innen. Auch System R (mit RStudio) und stylo ist durch die GUI zwar relativ einfach in der Bedienbarkeit, ist aber dennoch ein komplexes Verfahren; daher sollte man sich zunächst intensiv einarbeiten und anhand von Beispiel-Anwendungen ›üben‹.

Eine Alternative für das Thema Erwerb von benötigten Fähigkeiten aus beiden Disziplinen wäre, Studiengänge an den Hochschulen einzurichten, die eine kombinierte Ausbildung zur DH, also zu Informatik **und** Geisteswissenschaften anbieten. Mit der reinen Addition klassischer Fächer aus beiden Disziplinen ist es in der Regel nicht getan, vielmehr sollten sie miteinander verzahnt werden. Das kann u.a. so aussehen, dass der Anwendungsteil (also die praktischen Übungen) der Informatikfächer aus dem Bereich der Literaturtextanalyse gewählt wird. System R mit RStudio und stylo, Python und KI-Programme mit allen Lernarten sollten auf jeden Fall dazugehören, ein Mindestanteil Statistik wäre eine sinnvolle Ergänzung.

Solche kombinierten bzw. interdisziplinären Studiengänge (Stichwort »Bindestrich-Informatik«) gibt es schon lange. Beispiele sind Wirtschaftsinformatik, Ingenieursinformatik, Rechtsinformatik, Elektro- und Informationstechnik, medizinische Informatik, Bio-Informatik, Data Science und Business Administration mit Informatik. Aktuell gibt es »über 1300 Studiengänge im Fachbereich Informatik, die an mehr als 260 Hochschulen in Deutschland angeboten werden. Dazu kommen noch einmal knapp 700 interdisziplinäre Informatikstudiengänge.«[90]

90 https://www.get-in-it.de/magazin/studium/trendstudium-bindestrich-informatik [Zugriff: 08.05.2023]. Zu den geeigneten Inhalten eines DH-Studiengangs siehe den betr. Artikel auf dem Download Server.

Abkürzungsverzeichnis

API Application Programming Interface
csv Character Separated Value
DVj Deutsche Vierteljahrszeitschrift für Literaturwissenschaft und Geistesgeschichte
GNU General Public License
GUI Graphical User Interface
i.d.R. in der Regel
KI Künstliche Intelligenz
KWIC Keywords in Context
LDA Latent Dirichlet Allocation
LIWC Linguistic Inquiry and Word Count
MFW Most Frequent Words
ML Machine Learning
NER Named Entity Recognition
NLP Natural Language Processing
NN Neuronale Netze
OCR Optical Character Recognition
OOP Objektorientierte Programmierung
PC Personal Computer
SSD Solid State Drive
TEI Text Encoding Initiative
txt Text(-Format)
UML Unified Modelling Language
URL Uniform Resource Locator
WTE Worttrennelement
WWW World Wide Web
XML Extensible Markup Language

Abbildungsverzeichnis

Literaturverzeichnis

Blei, D. M., Andrew, Y. Ng, Michael I. Jordan (2003): Latent Dirichlet Allocation. In: Journal of Machine Learning Research 3, S. 993–1022.

Blei, D. M. (2012): Probabilistic Topic Models. In: Communications of the ACM 55 (4), S. 77–84.

Büttner, A. u.a. (2017): »Delta« in der stilometrischen Autorschaftsattribution. In: Zeitschrift für digitale Geisteswissenschaften. PDF Format ohne Paginierung. Als text/html abrufbar unter DOI: 10.17175/2017_006.

Burrows, J. (2002): ›Delta‹: A Measure of Stylistic Difference and a Guide to Likely Authorship. In Literary and Linguistic Computing 17 (3), S. 267–287.

Chollet, F. (2018): Deep Learning mit Python und Keras, Das Praxis-Handbuch © des Titels »Deep Learning mit R und Keras« by mitp Verlags GmbH & Co. KG, Frechen. URL: https://www.mitp.de/out/media/9783958458932_Leseprobe.pdf [Zugriff: 25.10.2022].

Claus, V. u.a. (2006): Duden Informatik A – Z: Fachlexikon für Studium, Ausbildung und Beruf, 4. Aufl., Mannheim.

Dämon, K. (2022): Studie Digitalisierung und Arbeitsplätze, welche Jobs betroffen sind, Wirtschaftswoche 21./22. Sept. 2022, Seite 2f. in URL: https://www.wiwo.de/erfolg/beruf/studie-digitalisierung-und-arbeitsplaetze-welche-jobs-betroffen-sind/12724850-2.html [Zugriff: 25.10.2022].

Femmer, H. u.a. (2017): Rapid quality assurance with requirements smells. Journal of Systems and Software 123, S. 190–213.

Fischer, F. u.a. (2019): Programmable Corpora: Introducing DraCor, an Infrastructure for the Research on European Drama. In Proceedings of DH2019: »Complexities«, Utrecht University, doi:10.5281/zenodo.4284002

Fischer, F. u.a. (2022): GerDraCor, Drama Corpora Project, URL: https://dracor.org/ [Zugriff: 25.10.2022].

Fischer, P., Hofer, P. (2008): Lexikon der Informatik, 14. überarbeitete Auflage, Springer-Verlag Berlin Heidelberg.

Flüh, M. (2019): »Sentimentanalyse«. In: forTEXT. Literatur digital erforschen. URL: https://fortext.net/routinen/methoden/sentimentanalyse [Zugriff: 07. August 2023].

Gius, E., Jacke, J. (2015): Informatik und Hermeneutik. Zum Mehrwert interdisziplinärer Textanalyse. In: Grenzen und Möglichkeiten der Digital Humanities. Hg. von Constanze Baum/Thomas Stäcker (=Sonderband der Zeitschrift für digitale Geisteswissenschaften, 1). text/html Format. DOI: 10.17175/sb001_006 [Zugriff: 30.10.2023].

Haufe Online Redaktion (2022): Agile Methoden und Techniken im Überblick, 07.01.2022, URL: https://www.haufe.de/personal/hr-management/agile-methoden-definition-und-ueberblick_80_428832.html [Zugriff: 25.10.2022].

Homburg, T. u.a. (2020): Diskussionsbeitrag – Handreichung zur Rezension von Forschungssoftware in den Altertumswissenschaften/Impulse, URL: https://research-squirrel-engineers.github.io/Impuls_SoftwareRezensionen_DGUF/Draft.html [Zugriff: 09.08.2023].

Horstmann, J. (2018): Topic Modeling. In: forTEXT. Literatur digital erforschen. URL: https://fortext.net/routinen/methoden/topic-modeling [Zugriff: 30. Juli 2023].

Horstmann, J. (2019a): Analyse und Visualisierung mit CATMA. In: forTEXT. Literatur digital erforschen. URL: https://fortext.net/routinen/lerneinheiten/analyse-und-visualisierung-mit-catma [Zugriff: 30. Juli 2023].

Horstmann, J. (2019b): Stilometrie mit Stylo. In: forTEXT. Literatur digital erforschen. URL: https://fortext.net/routinen/lerneinheiten/stilometrie-mit-stylo [Zugriff: 02. August 2023].

Horstmann, J. (2019c): Stylo. In: forTEXT. Literatur digital erforschen. URL: https://fortext.net/tools/tools/stylo [Zugriff: 05. Februar 2024].

Horstmann, J. (2020): DraCor: Drama Corpora Project. In: forTEXT. Literatur digital erforschen. URL: https://fortext.net/ressourcen/textsammlungen/dracor-drama-corpora-project [Zugriff: 30. Juli 2023].

Jacke, J. (2018): Manuelle Annotation. In: forTEXT. Literatur digital erforschen. URL: https://fortext.net/routinen/methoden/manuelle-annotation [Zugriff: 02. August 2023].

Jannidis, F. u.a. (2021): Disruptionen der Literaturwissenschaft am Beispiel der DVjs in Fabrikation von Erkenntnis, Experimente in den Digital Humanities, Melusina Press.

Leopold, E. (2022): Zipfsches Gesetz, URL: https://www.cis.uni-muenchen.de/~micha/kurse/statistik-SS2009/wiederholung/Zipfkurz.pdf [Zugriff: 30.07.2023].

Lück, C. (o.J.): XML und Textkodierung. Mit einer Einführung in die Programmierung für Geisteswissenschaftler*innen. Studienbrief der FernUniversität in Hagen, Fakultät für Kultur- und Sozialwissenschaften, Hagen.

Marcus, S. (1973): Mathematische Poetik, Bukarest.

Meister, C. (2016): Dreidimensionale dynamische Daten-Visualisierung und Exploration für Digital Humanities-Forschung, URL: https://jcmeister.de/projects/3dh/ [Zugriff: 28.10.2022].

Pfister, M. (2004): Das Drama, Paderborn, S. 236–240.

Royce, W.W. (1970): Managing the Development of Large Software Systems. Proceedings of IEEE WESCON, 26, S. 328–388.

Schumacher, M. (2019): »CATMA«. In: forTEXT. Literatur digital erforschen. URL: https://fortext.net/tools/tools/catma [Zugriff: 30. Juli 2023].

Schumacher, M. (2022): »Lebe lieber literarisch«, URL: https://lebelieberliterarisch.de/tutorial-wie-ich-catma-und-stanford-ner-zusammen-nutze/ [Zugriff: 27.10.2022].

Schumacher, M., Vauth, M. (2022, § 1): »CATMA-Annotationen auswerten, Gold Standard erstellen und Inter-Annotator-Agreement berechnen mit GitMA«. In: forTEXT. Literatur digital erforschen. URL: https://fortext.net/routinen/lerneinheiten/catma-annotationen-auswerten-gold-standard-erstellen-und-inter-annotator [Zugriff: 29. Oktober 2023].

Schumacher, M., Becker, K. (2021): »Lyra«. In: forTEXT. Literatur digital erforschen. URL: https://fortext.net/tools/tools/lyra [Zugriff: 09. August 2023]. Solomon, M. (1973): Mathematische Poetik. Bukarest: Ed. Academiei, 1973, S. 287–370.

Thamm, A. (2023): »Word2vec«; URL https://www.alexanderthamm.com/de/data-science-glossar/word2vec/ [Zugriff: 30. Juli 2023].

Weitin, T. (2021): Digitale Literaturgeschichte, Springer Berlin Heidelberg 2021, Verlag J.B.METZLER.

Weyrauch, A. (2015): Wie wahrscheinlich ist es, dass ich durch einen Computer ersetzt werde? In Süddeutsche Zeitung 2015, S. 1, URL: https://gfx.sueddeutsche.de/pages/automatisierung/ [Zugriff: 09.08.2023].

Anhang

Pseudocode und Programm zur Primzahlerkennung

Beispiel: Programm zur Prüfung einer Zahl n auf Primzahl mit Probedivisionsmethode. Mit dieser Methode wird die zu prüfende Zahl n sukzessive durch alle ganzen Zahlen von 2 bis n-1 (bzw. bis zur Wurzel aus n reicht auch) dividiert. Falls dabei ein Rest = 0 vorkommt, dann ist n keine Primzahl, sonst schon.
Hier der Pseudocode und je ein Java- und ein Python-Programm inkl. Funktion dazu:

```
Eingabe:
   Eine ganze Zahl n (ab 2 aufwärts; bei n = 0 Ende)
Verarbeitung:
   Setze Boolsche Variable istPrimzahl auf wahr (also zunächst Annahme: n ist eine Primzahl)
   Für eine Zahl i, die von 2 bis n (bzw. bis Wurzel aus n) aufsteigend in 1er Schritten läuft, prüfe
      Wenn der Rest der Division von n durch i größer 0 ist
         dann ist n keine Primzahl, also setze istPrimzahl auf falsch      # weil n durch dieses i teilbar
Ausgabe:
   Wenn istPrimzahl = wahr
      Anzeige "Die Zahl ist eine Primzahl"
   Sonst
      Anzeige "Die Zahl ist keine Primzahl"
```

Das vereinfachte **Java-Programm** für die Probedivision kann wie folgt aussehen:

```
public class Primzahl {
  public static void main(String[] args) {
    long n = 7;                        // Platzhalter für die Benutzereingabe
    boolean istPrimzahl = true;        // Initialisierung
    for (int i = 2; i*i < n; ++i)      // Schleife
       if (n % i == 0) {
         istPrimzahl = false;
         break;                        // break ist eigentlich kein guter Programmierstil
       }
    if (istPrimzahl == true)
      System.out.println("\n" + n + " ist Primzahl");   // Platzhalter für die Ausgabe
    else
      System.out.println("\n" + n + " ist keine Primzahl");
  }
}
```

Alternative kann eine Funktion pruefe() genutzt werden:

```
public static boolean pruefe(long n) {
   boolean istPrimzahl = true;
   for (int i = 2; i*i < n; ++i)
      if (n % i == 0)
        {istPrimzahl = false; break;}
    return istPrimzahl;      // Rückgabewert der Funktion
}
```

Dann ruft das Hauptprogramm diese Funkton wie folgt auf:

```
   istPrimzahl = pruefe(n);        // die Funktion pruefe() liefert true oder false zurück
```

Vereinfachtes **Python-Programm** mit Funktion:

```
import math
def istPrim(zahl):    # Funktion
    istPrimzahl = True
    for i in range(2, int(math.sqrt(n))):
        if n % i == 0:
            istPrimzahl = False
            break
    return(istPrimzahl)

# Hauptprogramm
while True:   # Dauerschleife bis Eingabe 0
    n = int(input("Zahl (0 = Ende): "))
    if n == 0:
        break
    istPrimzahl = istPrim(n)   # Aufruf der Funktion istPrim()
    if n > 1 and istPrimzahl:
        print("\n" + str(n) + " ist Primzahl")
    else:
        print("\n" + str(n) + " ist keine Primzahl")
```