Verzweigen, Kopieren, Verschmelzen Mediale Praktiken kollektiver Autorschaft

Markus Krajewski, Universität Basel,

local version 0.7:2019-03-26, github.com/nachsommer/VersionsKontrolle: a83e87-dirty

erscheint in: Daniel Ehrmann, Thomas Traupmann (Hrsg.), Kollektive(s) Schreiben, Wilhelm Fink Verlag, Paderborn 2019

Inhaltsverzeichnis

Die Geschichte verteilter Autorschaft lässt sich unschwer als die lange Genealogie des Schreibens überhaupt identifizieren, steht doch an den Anfängen der Schrift kein Individuum, kein Autor im Sinne seiner goethezeitlichen Ausdifferenzierung zur männlichen Verkörperung einer Genie-Ästhetik, sondern immer schon ein Kollektiv, sei es nun, um den Pentateuch zu verfassen oder das Gilgamesch-Epos, sei es, um unter einem mutmasslichen Kollektivsingular wie Homer nachträglich dessen gesungene Verse auf Papyrus zu fixieren. Texte sind *qua definitionem* Verbindungen loser Fäden, die von mehr als einer Person bearbeitet werden. Dies gilt umso mehr, wenn man einen die Philologie übersteigenden Textbegriff zugrunde legt, der ebenso operative Schriften, also Code und seine Entwicklung in Softwareprojekten, betrachtet, die inzwischen – schon infolge ihrer Komplexität – zumeist von mehr als einer Person vorangetrieben werden.

Zu den aufwendigsten und komplexesten Software-Projekten zählt die Entwicklung von Betriebssystemen, die sowohl im kommerziellen Kontext (Windows, macOS) als auch bei freier Software (Unix, Linux) eine Vielzahl von Mitarbeitern und deren unterschiedlichste Beiträge zu koordinieren haben. Dass die Entwicklung eines derart umfassenden Projekts keineswegs nur linear erfolgt, dass also auf Version 10.11 erwartungsgemäss 10.12 folgt, wird unmittelbar einsichtig, wenn man etwa die Genealogie von verschiedenen Unix-Derivaten verfolgt, wo aus dem »Unnamed PDP-7 operating system« von 1969 so unterschiedliche Systeme wie Solaris, BSD, HP-UX, macOS oder eben Linux hervorgegangen sind (Abb. ??)

Eine ähnliche, in sich jedoch noch ungleich verzwicktere, gelegentlich auch mehrere Zweige verschmelzende Entwicklung zeigt sich, wenn man allein die interne Entwicklung von Linux betrachtet (vgl. Abb. ??, links, zweite grüne Säule). Diese erweist sich als beinahe schon darwinistisches, von zahlreichen Bifurkationen geprägtes evolutionäres Schema. (Abb. ?? auf S. ??). Jede dieser zur Gegenwart strebenden Linien repräsentiert Tausende, oftmals Millionen Zeilen Code, die allesamt in einem fein abgestimmten, seinerseits über rund 50 Jahre entwickelten Milieu von kollektiver Autorschaft entstanden sind. Abb. ?? stellt damit einen regelrechten Wissensbaum dar, der die verschiedenen Derivate, also Abspaltungen von frei entwickelten Betriebssystemen bzw. Paketzusammenstellungen (Distributionen) aus den ursprünglichen, von Richard Stallmann seit 1983 entwickelten GNU-Distribution heraus dokumentiert. Aus Stallmanns Projekt gingen später weitere bekannte Erweiterungen und Neuansätze hervor, wie etwa das 1992 von Linus Torvalds veröffentlichte Linux, das wiederum selbst in zahlreichen unterschiedlichen Distributionen angeboten wird, von denen Debian, Slackware oder Red Hat nur die bekanntesten sind; als vergleichsweise junge Entwicklung zählt auch Android samt seiner Derivate dazu (ab 2007, in Abb. ?? am unteren Bildrand). Manche der Zweige (ver-)enden zu einem gewissen Zeitpunkt. Andere gehen in einem weiteren Projekte auf (gestrichelte Linien, etwa von Solus OS zu

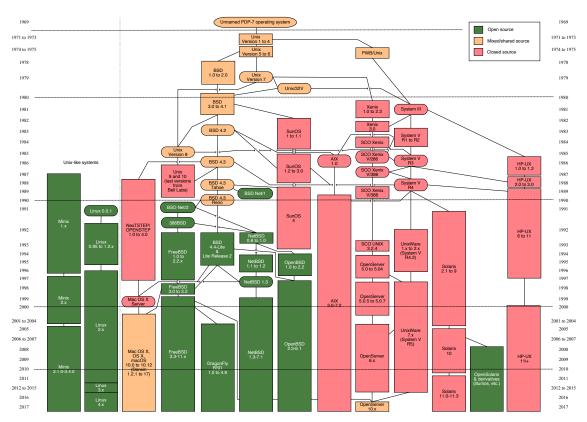


Abbildung 1: Genealogie diverser Unix-Derivate

Evolve OS). Die zahlreiche lose Enden am rechten Rand des Diagramms repräsentieren dabei jedoch nicht nur die Gegenwart des Schreibens, sondern auch die vielen möglichen Zukünfte von kollektiver Autorschaft. Denn sobald mehrere, wenn nicht gar tausende Entwickler parallel an ein und demselben Code arbeiten, erfordert es besonderer Maßnahmen, um die Konsistenz dieser kollaborativen Autorschaft sicherzustellen.

In den folgenden, mit arabischer Zählung numerierten Abschnitten soll zunächst ein Blick darauf geworfen werden, wie grossangelegte kollektive Schreibprojekte mit Hilfe von sog. Versionsverwaltungen und ihren grundlegenden Befehlen wie etwa *branch* und *merge* medientechnisch organisiert sind, bevor es im letzten Abschnitt Coda mit einem Blick auf die sog. Auto(r)Korrektur noch darum geht einen Weg aufzuweisen, wie die historischen Befunde der Kultur- und Literaturgeschichte für die Gegenwart der gemeinsamen Schreibumgebungen produktiv gemacht werden können. Dabei geht es nicht allein darum zu sehen, welchen Befehlssätzen und Befehlsketten die Autoren und ihre Codezeilen dabei gehorchen. Vielmehr geht es ebenso darum, die informatischen Routinen auf ihre historischen Grundlagen hin zu lesen, um mit diesem Blick auf die Geschichte verteilter

Autorschaft die kulturtechnische Funktionsweise kollaborativen Schreibens herauszuarbeiten. Mithin geht es um die Absicht, der Geschichtsvergessenheit der *software studies* entgegenzuarbeiten, um die gegenwärtige Praktik kollektiver Autorschaft ihrerseits in ihren Entwicklungslinien zurück zu verfolgen, nicht zuletzt geleitet von dem Anspruch, die informatischen Praktiken nicht einfach als immer schon gegeben hinzunehmen, sondern sie selbst durch eine vorgängige historische Entwicklung zu bereichern. Diese Rekonstruktion stützt sich dabei zunächst auf einige exemplarische Geschichten beziehungsweise historische Szenarien, die sich mehr oder minder synchron ereignen, und zwar um 1780, zum einen im Frankreich des Ancien Regime, zum zweiten in der Hauptstadt des Heiligen Römischen Reichs Deutscher Nation, in Wien, und zum dritten, in Weimar mit seiner goethezeitlichen Perfektionierung von Autorschaft, wobei sich freilich immer auch ganz andere Szenarien, etwa in London 1916 und anderenorts, anführen liessen.

1 Versionierungen – A Brief Introduction

Die eigentümlichen Imperative *branch* und *merge* sind nicht nur unschuldige (Stamm-)Formen englischer Verben, sondern finden ebenso in einem informatischen Kontext Verwendung, konkret bei der sogenannten *version control*. Derartige Versionsverwaltungen kommen einerseits im Hintergrund von organisatorischen Maßnahmen auf Betriebssystemen zum Einsatz, also etwa bei der eingebauten Backup-Funktion auf macOS namens >TimeMachine<. Andererseits bilden sie das Kernstück sowohl für lokale als auch für zentrale oder gar global verteilte Softwareentwicklungsprojekte, wo Entwickler an unterschiedlichen Orten gleichzeitig an demselben Code arbeiten, dessen Änderungen demzufolge zeichengenau protokolliert werden und nachvollziehbar bleiben müssen. Das bekannteste System dieser Art dürfte derzeit die von Linus Torvalds initiierte Platform *github* sein, auf der unter github.com/nachsommer/VersionsKontrolle auch eine digitale Version des vorliegenden Texts zu finden ist.

Das Ziel der Codeentwicklung ist dabei – leicht idealisiert – wie bei einer konventionellen Textproduktion zu verstehen, bei der am Ende eine Abfolge von Zeichen entsteht, die – bei hinreichendem Interesse möglicher Leser und ausreichender Schreibkunst der Autoren – vom ersten bis zum letzten Zeichen rezipiert wird. Ähnlich akribisch kann man sich im informatischen Kontext als zentrale Entscheidungsgewalt den sog. *compiler* vorstellen, also jene Instanz bei der Programmierung, die den in einer beliebigen (höheren) Programmiersprache vorbereiteten Algorithmus in den allein ausführbaren Code der Maschinensprache übersetzt. Auch bei diesem Übersetzungsprozess wird jedes Zeichen, jeder Befehl, jede Schlei-

^{1.} yuill:2008.

fe, jede Datenstruktur sequentiell eingelesen, validiert und interpretiert. Man muss sich den *compiler* als einen Meister des *close reading* vorstellen.² Um also einen Programmcode lauffähigk zu machen, muss er einmal linearisiert, das heisst jeder der zahlreichen Befehle muss Zeile für Zeile ausgewertet werden. Der *compiler* zieht die Teile des Programmcodes aus unterschiedlichsten Bereichen zusammen, aus entlegenen Programmbibliotheken ebenso wie aus offenen Quellen, die dezentral im Internet in entsprechenden Repositorien vorgehalten werden, um alles in eine lineare Abfolge zu bringen.

Nun kann es jedoch vorkommen, dass über die genaue Abfolge der Befehle, Programm- und Datenstrukturen Uneinigkeit herrscht innerhalb der Gemeinschaft der Codeentwickler eines bestimmten Projekts. Angenommen, ab Codezeile 13'531 stehen folgende Befehle:

```
leseBrief("Cécile Volanges", "Sophie Carnay",
13531
       gegeben("Paris",1781-08-03));
                                                            // 1. Brief
13532
13533
     leseBrief("Marquise de Merteuil", "Vicomte de Valmont",
                                                            // 2. Brief
       gegeben("Paris",1781-08-04));
     leseBrief("Cécile Volanges", "Sophie Carnay",
13537
       gegeben("Paris",1781-08-04));
                                                            // 3. Brief
13538
13539
13540 leseBrief("Vicomte de Valmont", "Marquise de Merteuil",
       gegeben("Schloß Cormatin",1781-08-05));
                                                              // 4. Brief
13541
13542
13543 leseBrief("Marquise de Merteuil", "Vicomte de Valmont",
     gegeben("Paris",1781-08-07));
```

Weiter angenommen, dass bezüglich der Codezeile 13'541 eine Diskussion in der weltweiten Entwicklergemeinschaft entbrennt, weil der Entwickler Egmont der Meinung ist, hier müsse statt >Schloß Cormatin< vielmehr >Schloß Chambord < stehen, da infolge eines Unwetters im Südosten von Paris die Straßen am 6. August 1781 nach Burgund unwegsam waren, so dass kein Postillon seine Briefe zustellen konnte, was wiederum dazu führte, dass infolge der üblichen Brieflaufzeiten die Antwort der Marquise de Merteuil niemals am 7. August hätte geschrieben werden können. Das Programm sei also, so Egmont, an dieser Stelle fehlerhaft. Der Entwickler Richard, auf den der ursprüngliche Code zurückgeht, kann sich mit diesem Argument allerdings nicht einverstanden erklären und beharrt auf seiner anfänglichen Lokalisierung des Briefs des Vicomte de Valmont auf Schloß Cormatin. Der Konflikt bleibt ungelöst und der Programmcode wird an dieser Stelle kurzerhand verzweigt, das heisst mit Hilfe des Befehls branch gelingt es, den Code zu duplizieren, um beide Varianten parallel zueinander existieren zu lassen (Abb. ??). Egmont schert also an dieser Stelle aus dem linearen Ablauf der Befehlskette aus, indem er einfach

^{2.} krajewski+vismann:2009.

seine eigene Variante unter dem Etikett eines neuen *branch* (Zweigs) der Allgemeinheit zur Verfügung stellt.

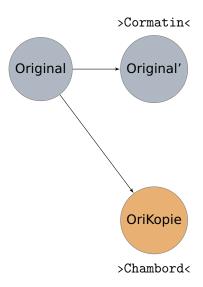


Abbildung 2: Varianten einer Entwicklung

Beide Stränge sind nach wie vor für alle Entwickler sichtbar und nahezu identisch, bis auf die kleine Abweichung von Egmont, der seinen source code gegenüber dem zwischenzeitlich womöglich ebenfalls weiter entwickeltem Original' von Richard nach den eigenen Vorstellungen abgeändert hat. Um hier den Überblick nicht zu verlieren, kann man mit dem kleinen Hilfsprogramm diff den Befehl erteilen, sich die Unterschiede in den beiden Quellcodes anzeigen zu lassen (Abb. ??).

Abbildung 3: diff von Original und Kopie

diff macht also den kleinen Unterschied sichtbar. Das Programm hebt die Abweichungen zweier in weiten Teilen identischen Dokumente hervor, oder um es – leicht abweichend – mit einem informatischen Begriff zu sagen: diff macht die Deltas innerhalb des Codes sichtbar, oder um es – erneut leicht abweichend – mit einem philosophischen Begriff zu sagen: diff führt die différance zwischen Original und Kopie vor. – Das Programm diff wurde in den frühen 70er Jahren von Douglas McIlroy an den Bell Labs in New Jersey geschrieben. Die Denkfigur différance wurde in den frühen 70er Jahren von Jacques Derrida an der ENS in Paris entwickelt.

Wo wäre in der langen Geschichte kollektiver Autorschaft der historische Vorläufer zum diff zu verorten? Welche Instanz sorgt sich um etwaige Satz- oder Tippfehler, Zeilen-Unterschiede, Kopierfehler, unmerkliche, wenn nicht infinitesimale Details in der Wort(dar-)stellung? Kurzum, was ist das klassische Pendant zum diff und seiner Fixierung der Deltas? Ebenso kurz gesagt: Der Herausgeber oder Bearbeiter einer Edition, derjenige also, der für die Sicherung des Textes, für die Entscheidung, dieser und nicht der anderen Variante den Vorzug zu geben, verantwortlich zeichnet, wobei er freilich – etwa bei historisch-kritischen Editionen – die Kontingenz der Varianten ebenfalls zur Darstellung zu bringen hat. Im diff kondensiert – oder mit Blick auf den letzten Abschnitt: schmilzt – also eine lange Theoriegeschichte der Philologie und Editionswissenschaft.

Nun könnten sich beide Zweige von ein und demselben Programm parallel zueinander weiterentwickeln, sich dabei zunehmend unterscheiden, in mehr als nur einer Zeile voneinander abweichen, so lange bis es zwei ganz unterschiedliche Programme geworden sein werden. Doch die auseinanderstrebende Bewegung der beiden Teile wird in diesem (fiktiven) Beispiel durch einen neuen Forschungsstand jäh unterbunden: In einem Konvolut im Nachlass von Pierre-Ambroise-François Choderlos de Laclos taucht der Hinweis auf, dass es sich bei dem von ihm in der Druckfassung bewusst als leere Variablen belassenen Ortsangaben von Valmonts Aufenthaltsort um das Schloß Bussy-Rabutin gehandelt habe, wie der Nutzer Oliva glaubhaft machen kann. Der Konflikt ist damit durch eine neue archivalische Evidenz geschlichtet, die beiden falschen Angaben >Chambord< und >Cormatin < gilt es zu ersetzen durch >Schloß Bussy-Rabutin <, um damit die beiden separaten Stränge wieder zusammenzuführen (Abb. ??). Diese Konvergenzbewegung wird durch den Befehl merge erreicht, der die beiden konkurrierenden Darstellungen wieder vereint, indem zunächst einer der beiden Versionen in Programmzeile 13'541 der Vorzug gegeben wird, um den Code dann mit der neuen Erkenntnis und ergänzt um einen Kommentar erneut zu einem einzigen Zweig zu fusionieren. Dieses Verfahren, das auch als three-way-merge bezeichnet wird, erfreut sich nicht nur in der theoretischen Informatik der Gegenwart einer regen Forschungstätigkeit, sondern dürfte philologisch Gebildeteten nicht ganz unbekannt vorkommen. Stellt es doch eine recht alltägliche Problematik bei der Herstellung

historisch-kritischer Editionen dar, wo ebenso zwischen verschiedenen Varianten eines Texts zu differenzieren und anschliessend eine Entscheidung zu fällen ist, welcher Variante der Vorzug zu geben sei.

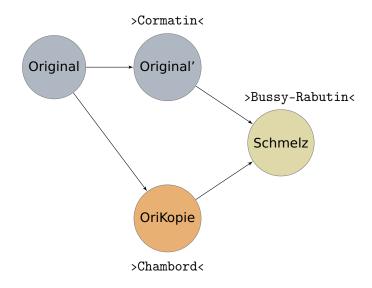


Abbildung 4: Neue Variante der Varianten

Auf die medialen Praktiken *branch* und *merge* sei nun etwas genauer eingegangen, um die Verfahren, wie sie in informatischen Kollaborationen derzeit weltweit Anwendung finden, auf ihre eigene Geschichte hin zu befragen. Denn ein besonderer Vorzug von Versionskontrollsystemen besteht darin, dass ein solches System stets reversibel in der Zeit bleibt, das heisst, man kann nahezu mühelos zwischen unterschiedlichen Zeitpunkten der Codeentwicklung wandern, die Zeitachse also bei Bedarf wie einen Schieber zurück bewegen, um jegliche Änderung am Code wieder ungeschehen zu machen. Hier zeigt sich auch schon eine wichtige Differenz: Weder in der Historiographie von Software noch in der Geschichtsschreibung von Kulturtechniken wie dem Kopieren besteht (bedauerlicherweise) die Möglichkeit einer Rückkehr zum *status quo ante*. Es sei denn, man bewegt sich immersiv durch ein spezifisches Medium in andere Welten. Und dieses Medium ist die Fiktion.

2 Verzweigen

Der unscheinbare Befehl *branch* bewirkt nichts weniger, als mit einem einzigen schlichten Kopiervorgang ein Paralleluniversum zu erzeugen. Der gesamte Kosmos des fiktiven Softwareprojekts [*Gefährliche Liebschaften*] wird mit all seinen Routinen, Nischen, Fehlern, Kommentaren, Besonderhei-

ten und Unzulänglichkeiten leichterhand dupliziert, um sodann in einem kleinen Detail verändert zu werden. Der Zeitpunkt der Befehlserteilung branch markiert den Bifurkationspunkt, ab dem sich die Welten teilen, um fortan zwei parallele Weltläufte mit zwei »verschiedenen Zukünften«³ zu generieren. Was dem einen als ein »wirrer Haufen widersprüchlicher Entwürfe« erscheint, ist für den anderen ein wohlgeordnetes »Labyrinth aus Symbolen«.⁴ Oder noch genauer: ein »Labyrinth aus Zeit«.⁵ Denn diese Struktur aus parallelen Welten mit ihrer unbegrenzten Möglichkeit zur Kontingenz »erschafft so verschiedene Zukünfte, verschiedene Zeiten, die ebenfalls auswuchern und sich verzweigen.«⁶

Die Erzählung von Jorge Luis Borges von 1941, aus der die Zitate im vorangehenden Absatz stammen, entwickelt eine Struktur, die jener eigenartigen textuellen Parallelwelt entspricht, die durch einen branch-Befehl erzeugt wird. Wie in dem Briefroman von Choderlos de Laclos gibt es in dieser Erzählung eine Rahmenhandlung, in der ein fiktiver Herausgeber dem Leser von einem unverhofft gefundenen Text-Fragment berichtet, das eine zuvor noch rätselhafte Kriegshandlung erhellt. In diesem Fragment wird berichtet, wie der Ich-Erzähler, ein chinesischer Spion, der im Ersten Weltkrieg in England für die Deutschen kundschaftet, den britischen Sinologen Stephen Albert aufsucht. Bei diesem Besuch begegnet der Chinese einer besonderen Struktur, und zwar dem von einem seiner Vorfahren entworfenen »Garten der Pfade, die sich verzweigen«, – so der Titel der Erzählung. Dieser labyrinthische Garten erstreckt sich jedoch nicht im Raum, sondern in der Zeit, insofern er aus einem in zahlreichen Versionen durchgespielten Roman besteht, der – wie in Leibniz Theodizee⁷ – alle möglichen Begebenheiten in parallelisierten Varianten enthält. Der Autor dieses Romans »glaubte an unendliche Zeitreihen, an ein wachsendes, schwindelerregendes Netz auseinander- und zueinanderstrebender und paralleler Zeiten. Dieses Webmuster aus Zeiten, die sich einander nähern, sich verzweigen, sich scheiden oder einander jahrhundertelang ignorieren, umfaßt alle Möglichkeiten.«⁸ Wie sich diese Problematik von Unendlichkeit materiell bewerkstelligen lässt, wird in der Erzählung freilich ebenso reflektiert, nämlich entweder zyklisch, so wie es Vladimir Nabokov mit seinem Karteikarten-Roman Pale Fire zwei Jahrzehnte später vorführt, oder rekursiv, wie Scheherazade, die an einer bestimmten Stelle in Tausendundeiner Nacht aus einer bestimmten Stelle von Tausendundeiner Nacht vorliest.

^{3.} borges:1941. 4. borges:1941. 5. borges:1941. 6. borges:1941.

^{7.} Die Konstruktion gleicht in gewisser Weise der Konstellation, die Leibniz in seiner Theodizee-Problematik durchspielt: Die göttliche Ordnung kennt zahlreiche Welten, die nebeneinander bestehen, sich gleichen, aber doch in einigen signifikanten Details unterscheiden. Allerdings sind im Fall der Software keine Qualitätskriterien wie gut und böse ausschlaggebend, sondern eher der Zuspruch und die Nutzung durch andere Entwickler. Zudem führt die Parallelität der beiden Code-Ordnungen (um nicht Kosmoi zu schreiben) vor allem die Kontingenz vor Augen, dass jede artifizielle Welt auch anders sein könnte.

^{8.} borges:1941.

Es ist ein »Labyrinth, das die Vergangenheit umfaßte und die Zukunft [...] Zum Beispiel kommen Sie in dieses Haus, aber in einer der möglichen Vergangenheiten sind Sie mein Feind gewesen, in einer anderen mein Freund.«⁹ Und wie die Geschichte dann weiter zeigt, wird er auch beides zugleich gewesen sein – ganz so wie die Version *Original* neben der Version *Original*' existiert.

Es ist kaum nötig zu erwähnen, dass diese Charakteristik des unendlichen Romans, die Borges entwirft, eine ziemlich präzise strukturelle Beschreibung dessen liefert, was eine Software-Versionsverwaltung rund 50 Jahre später bereitstellt: >ein wachsendes, schwindelerregendes Netz auseinander- und zueinanderstrebender und paralleler Zeiten<, in dem die verschiedenen Varianten eines Texts nebeneinander, voneinander unabhängig mit jeweils eigenen, offenen Entwicklungshorizonten entstehen. Mit anderen Worten, die vergleichsweise kurze Geschichte von Software und ihrer jeweiligen Maßnahmen, Kontrolle über die Quellen zu erlangen oder zu erhalten, korrespondiert – zumindest untergründig – mit der langen Geschichte von literarischer oder gar kollektiver Autorschaft. Man könnte Borges abgründige Geschichte über die »Verzweigung in der Zeit«¹⁰ daher als eine Art literarische Präfiguration der informatischen Versionskontrolle verstehen.

Bevor nun im weiteren Textverlauf von der medialen Praktik des Verzweigens seinerseits wieder verzweigt wird das Verschmelzen, sei noch ein kleiner Unterzweig eingebunden, der so etwas wie die fundamentale Übertragungsfunktion dieser Verfahren darstellt. Denn weder *branch* noch *merge* kommen ohne einen Vorgang aus, der die Daten von A nach B oder nach A' schafft; der sie prüft, validiert, transferiert und dupliziert. Es folgt demnach eine kurze Verzweigung zum Vorgang des Kopierens.

3 Kopieren

Wenn die lange Geschichte des (literarischen) Schreibens über weite Strecken und Genealogien keineswegs das Geschäft einer Einzelperson ist, sondern vielmehr Teamwork, wenn das Schreiben also in den überwiegenden Fällen eine kollektive Tätigkeit darstellt, dann kommt dem Abschreiben oder Kopieren dabei eine besondere Funktionsstelle zu. Zum einen, weil das rasche Vervielfältigen von Texten – vor dem Zeitalter der technischen Reproduzierbarkeit durch Photographie oder Photokopie – nicht selten im Modus des Diktierens stattfindet, hier also zwei interagierende Personen schriftliche Artefakte über das Medium der Stimme wieder in schriftliche Artefakte transformieren. Zum anderen, weil selbst beim Abschreiben beispielsweise einer Bibelpassage durch einen einzelnen Mönch im Skriptorium ebenfalls zwei Personen interagieren, insofern die schriftlich fixierte

^{9.} borges:1941. 10. borges:1941.

Rede eines Abwesenden das Original darstellt, das durch das Medium des anwesenden Schreibers in Kopie dupliziert wird.

Nun mag man einwenden, dass für die massenhafte Vervielfältigung von Texten seit dem 15. Jahrhundert mit dem Buchdruck ein Medium bereitsteht, dass diese komplizierten Vergegenwärtigungen von Text über Stimme und Handschrift eigentlich überflüssig macht. Das trifft zweifellos zu auf Kopiervorgänge, bei denen ausreichend Zeit zur Anfertigung der Kopien bereit steht. In Situationen allerdings, wo es auf einzelne Minuten ankommt, wo Zeit kritisch, weil knapp wird oder der Aufwand zur Einrichtung einer Druckvorlage ohnehin viel zu gross wäre, weil es nur einer einzigen Abschrift bedarf, findet die bewährte Form der Vervielfältigung durch Diktat oder ein individuelles Duplikat ihren Einsatz.

Das folgende Szenario rekonstruiert die Funktionen einer hochprofessionalisierten, immer schon international arbeitenden Institution, in der das Vervielfältigen in Serie, der Akt des Kopierens, des Filterns und ggf. noch des Verschlüsselns zur exklusiven Aufgabe zählte. Der Schauplatz ist eine Institution, die unter wechselnden Namen wie >geheimbe Zyffer Weeßen«, >Zyffer Scretariat«, >Kabinets-Secretariat«, >Visitations- und Interceptions-Geschäft<, >Geheime Kabinets-Kanzlei< oder auf ihren französischen Ursprung unter Ludwig XIV. rekurrierend schlicht als cabinet noir bezeichnet worden ist. 11 Dieser auch als >Brief-Inquisition < bezeichneten Abteilung im Umfeld eines weltlichen Herrschers unterstand es, den gesamten Postverkehr eines Landes, insbesondere in der Residenzstadt zu überwachen, die wichtigen Briefe nicht nur abzufangen, sondern unbemerkt zu entsiegeln, zu öffnen, zu durchmustern, zu lesen, sie ggf. zu entschlüsseln, zu kopieren, zu registrieren, zu verschliessen, um sie endlich mit einem gefälschten Siegel zu versehen und anschliessend wieder dem ursprünglich beabsichtigten Postlauf zu übergeben.

Insbesondere in Wien, der Stadt des Kaisers, legt man viel Wert auf einen geräuschlosen Ablauf im Hintergrund des weitverzweigten Postwesens, das nicht zuletzt von der traditionellen Nähe der Habsburger zur Familie Thurn und Taxis bestimmt wird. Am Ende des 18. Jahrhunderts residiert diese Reichszentrale Intelligenz-Agentur in unmittelbarer Nähe zur Macht, vis-à-vis zur Wiener Hofburg, um ihrer speziellen Auffassung von Aufklärung nachzugehen:

Abends Schlag 7 Uhr schloß sich die Postanstalt und die Briefwagen schienen abzufahren. Sie begaben sich aber in einen Hof des kaiserlichen Palastes, woselbst schwere Thore sich sogleich hinter ihnen schlossen. Dort befand sich das Schwarze Kabinett, die Stallburg.

Da öffnete man die Briefbeutel, sortirte die Briefe und legte diejenigen bei Seite, welche von Gesandten, Banquiers und

^{11.} leeuw:1999.

einflußreichen Personen kamen. Der Briefwechsel mit dem Auslande zog meist ganz besondere Aufmerksamkeit auf sich. Die Siegel wurden abgelöst, die wichtigsten Stellen kopirt und die Briefe mit teuflischer Geschicklichkeit wieder verschlossen.¹²

Wenn selbst die wichtigsten Sendschreiben nicht lange verweilen dürfen, um noch in derselben Nacht auf den Weg ihrer eigentlichen Bestimmung gebracht zu werden, ist stets Eile geboten. Zwischen 80-100 Briefe schafft man täglich zu durchmustern bzw. zu perlustrieren – wie dieser Vorgang im Fachjargon heisst. Nach einer ersten Sichtung von Adressat und Absender, also einer Registrierung anhand der Meta-Daten, erfolgt bei Schreiben, die weitergehendes Interesse verheissen, eine genauere Autopsie des derart entwendeten Briefes, der »mittels einer sehr dünndochtigen brennenden Kerze mit >unruhiger< Hand aufgelassen und geöffnet [wurde]. Der Manipulant merkte sich schnell die im Kuvert liegenden Bestandteile, die Lage derselben und übergab das Briefpaket dem Subdirektor, der den Brief durchlas und entweder den ganzen Inhalt oder ihn auszugsweise kopieren ließ.«¹³ Nach einer ersten Übersicht der einzelnen Programmbestandteile, also einer Sichtung ihrer Lage, wird der Code weitergereicht, um zweitens, noch bei Bedarf entschlüsselt und dann interpretiert zu werden, bevor man ihn drittens, erneut arbeitsteilig zu kopieren sich anschickt. – Diese Operationskette verdient deshalb noch eigens Hervorhebung, weil ihre drei Schritte ebenso konstitutiv sind für den Vorgang des branching, dem sie notwendigerweise vorausgehen. Der eigentliche Kopiervorgang erfolgt sodann unter Einsatz einer medientechnisch verfeinerten ars dictaminis:14

Die Offiziale waren in der Regel Schnellschreiber. Hin und wieder gab es auch >short hand-Schreiber<. Man diktierte, um Zeit zu gewinnen. Zwei Offiziale nahmen einen Bogen und diktierten zwei Schnellschreibern, ja sogar vier Offiziale diktierten zugleich aus einem Bogen auf eine so geschickte Weise vier Kollegen, daß die Schreibenden nicht irre werden konnten. Auf diese Weise konnte ein Bogen in wenigen Minuten abgeschrieben werden. Im Notfalle kopierte das ganze Personal ohne Unterschied, Hofrat und Subdirektor mitinbegriffen.¹⁵

Der Kabinettsdirektor überprüft anschliessend diese derart im beschleunigten *multitasking* oder *parallel processing* gewonnenen Ergebnisse, filtert sie nach der jeweiligen politischen Interessenslage und reicht sie weiter direkt zum Kaiser bzw. zur Polizei. ¹⁶ Einen halben Briefbogen pro Minute kennzeichnet eine Datendurchsatzrate, die nicht so viel geringer bleibt als die einer *floppy disk* 200 Jahre später. Eine allfällige Verschlüsselung beansprucht um 1780 ebenfalls nur etwas mehr Zeit als um 1980. Es

^{12.} koenig:1875.

^{13.} **stix:1937**.

^{14.} krautter:1982.

^{15.} stix:1937.

^{16.} stix:1937.

kann daher nicht verwundern, wenn den Schwarzen Kabinetten schon im 19. Jahrhundert der Nimbus einer modernen, mit der neusten Medientechnik ihrer Zeit experimentierenden Intelligenzagentur konstatiert wird. »Sie glichen weniger Postämtern, als Laboratorien.« 17

Eine der entscheidenden Fähigkeiten, die in diesen Laboratorien stets geübt und weiter entwickelt werden, besteht in der Nachahmung der jeweiligen Handschriften. Kopieren bedeutet nämlich nicht nur, den Inhalt buchstabengetreu von einem Blatt auf das andere zu übertragen, sondern ebenso, sich des Stils, der Eigenheiten, der inneren wie der äusseren Form des Anderen im Brief - und nicht selten auch über den Brief hinaus anzuverwandeln. »Man öffnete die Briefe, schrieb sie ab und unterschob perfide Schreiben, in denen Handschrift, Schreibweise und Überschrift des Absenders mit wunderbarer Kunst nachgeahmt war«, fasst Emil König in seiner Streitschrift gegen die Verletzung des Briefgeheimnisses von 1875 diese Fähigkeit lakonisch zusammen. 18 Den Kopisten selbst schreibt König dabei eine derart exzessive mimetische Kraft zu, dass es nicht selten psychopathologische Züge annehme: »Nicht genug, daß sie die Briefe mit einer ganz erstaunlichen Gewandtheit öffneten und wieder versiegelten, ahmten sie auch die Schriftzüge nach, schrieben falsche Briefe, gaben falsche Rathschläge und betrogen Absender und Empfänger auf das Schändlichste. Ihre Arbeit erforderte übrigens eine so große Anspannung des Geistes, so viel Sorgfalt und Geschwindigkeit, dass mehrere dadurch den Verstand verloren.«19

Man muss nicht zwangsläufig verrückt werden oder in schändlicher Absicht arbeiten, wenn es gilt, sich mit einer spezifischen Form der high fidelity eines Anderen anzuverwandeln. Eine gesündere und ehrenvollere Form mimetischer Angleichung hinsichtlich der Briefkopien lässt sich zur selben Zeit in der Korrespondenzpraxis von Goethe beobachten, dessen Briefproduktion über die Jahrzehnte rund 20'000 Exemplare umfasst – trotz der expliziten Verweigerung des Geheimrats, die Feder selbst zu führen.

Wie hinlänglich bekannt ist das Aufschreibesystem 1800 keineswegs nur auf die Ausbildung neuer Dichter abgestellt, ²⁰ sondern bedient sich – auch und gerade bei Goethe – eines umfangreichen Apparates von Bedienten, also Sekretären, Schreibern, Kopisten, Kammerdienern und anderen Subalternen aller Art. Dabei ist besonders auffällig, dass alle Subalternen in spezifischer Weise eine Eigenart annehmen, die sie ihrem Herrn und Gebieter ähnlicher werden läßt. Johann Georg Paul Götze, der rund 17 Jahre bei Goethen dient, gelingt es etwa, sich die Handschrift seines Meisters zu solcher Perfektion anzueignen, dass selbst die Experten später bisweilen Mühe haben werden, sie vom Original treffsicher zu unterscheiden. Zudem übt Götze sich noch darin, zu zeichnen wie sein Vorbild. ²¹ Die Diener Geist und Stadelmann laufen dagegen mit einem anderen Wahrnehmungsfil-

^{17.} koenig:1875. 18. koenig:1875. 19. koenig:1875. 20. kittler:1995a.

^{21.} schleif:1965gay:1996.

ter ihres Herrn durch die Welt, oder genauer: in das Theater und durch Steinbrüche. »Alle Diener Goethes haben, jeder nach seinen Möglichkeiten, Züge des äußeren Gehabens ihres Herrn angenommen, sich seine Handschrift angewöhnt und aus seinen Wissensgebieten ihre Steckenpferde gewählt: [Philipp] Seidel philosophische, sprachliche und wirtschaftliche Themen, Geist Botanik, Stadelmann Geologie und Mineralogie, [Michael] Färber Osteologie.«22 Das hohe Maß an mimetischem Verlangen, der ungestillte Wunsch nach Anverwandlung, zeigt sich jedoch am prägnantesten bei Philipp Seidel, Goethes erstem Subalternen, der später dank seiner Verdienste zum Weimarer Kammerkalkulator befördert wird: »Er hatte sich ihm derart angeähnelt, daß sie ihn Goethes >vidimirte Kopie< nannten«.²³ Seidel versteht sich darauf, den Rededuktus seines Herrn, die Intonation ebenso beiläufig nachzuahmen wie gleich seinem Vorbild den Kopf zu schütteln und sogar dessen »Perpendikulargang« so täuschend echt zu imitieren, »daß man oft versucht war, ihn von weitem für Goethe selbst zu halten.«²⁴ Der Meister dupliziert sich in seinen Domestiken.

Es wäre irrig anzunehmen, dass Goethe seine Briefe selbst schreibt. Abgesehen vom in grosser Kanzleischrift geschwungenen G. als Signatur setzt er den schon im Original als Kopie durch die nachahmende Hand der Schreiber verfassten Briefe nichts weiter hinzu als gelegentliche Grüsse oder allfällige Addenda. 25 Seidel dient zudem auch als historisches Vorbild für den Briefeschreiber Richard in Goethes Egmont, der die Briefe seines Herrn in einem Akt exzessiver Mimesis auch inhaltlich für seinen Meister ausfertigt.²⁶ Erst wenn diese eng verflochtene, kollaborative Autorschaft einmal gestört ist, sieht sich G. noch genötigt, selbst zur Feder zu greifen, wenn auch nicht ohne Widerwillen. So klagt Goethe, als 1813 sein zusätzlich zum schreibkundigen Domestiken eingestellter Sekretär Ernst Carl Christian John einmal unpäßlich ist: »Seit vierzehn Tagen hat sich leider meine adoptive rechte Hand kranckheitshalber in's Bette gelegt und meine angebohrne Rechte ist so faul als ungeschickt, dergestalt daß sie immer Entschuldigung zu finden weis wenn ihr ein Briefblatt vorgelegt wird.«27

Über Goethes Praxis des Briefeschreibens und die rekursiven Bezugnahmen, wechselnden Autorschaften und unterschiedlichen Befehlsebenen im Zusammenspiel mit seinen Dienern gäbe es noch viel zu sagen. ²⁸ An dieser Stelle sei der Unterzweig zum grundlegenden *copy*-Befehl jedoch schon wieder verlassen zugunsten der zweiten heuristischen Praktik der Versionskontrolle, dem Verschmelzen. Sie behandelt jene Texte, die einstmals auseinander hervorgingen und zwischenzeitlich einseitig verändert wurden, um sie wieder zu einer neuen Version zu fusionieren.

 ^{22.} schleif:1965.
 23. schleif:1965.
 24. lyncker:1912.
 25. schleif:1965.

 26. krajewski:2010.
 27. goethe:1887.
 28. krajewski:2010schoene:2015.

4 Verschmelzen

Ein Buch ist eine Einheit, der immer schon die Zerlegung droht. Schliesslich bezeichnet ein analytischer Umgang mit dem enthaltenen Text nichts anderes, als diese Einheit wieder zu zerteilen, in seine Bestandteile wie etwa in ein zentrales Argument, das Dekorum, einzelne Gedanken, Thesen, Vermutungen, Exempla, weiterführende Literatur usw. aufzuspalten. Mit den so gewonnen Fragmenten oder derart ausgelösten Textelementen lässt sich dann weiterarbeiten, sei es mit direkten Zitaten oder indirekten Paraphrasen, in Form von Exzerpten für die eigene Textproduktion oder von allgemeinen Lektürenotaten für den späteren Gebrauch, sei es schlicht durch die Erinnerung an inspierende Textstellen oder systematisch durch eine Übernahme der Metadaten des Texts, also seine bibliographischen Angaben, zur Katalogisierung des neu gewonnenen Wissens.

Ein Buch ist aber auch eine Einheit, die ihrerseits immer schon aus Fragmenten aufgebaut ist.²⁹ Fliessen doch selbst in den kohärentesten, für sich stehenden Text immer schon vorgängige Bausteine ein wie bestimmte Formulierungen, Erzählmuster oder theoretische Begriffe, implizite Bezugnahmen oder unausgewiesene Inspirationen, explizit übernommene Denkfiguren oder wörtliche Zitate, die ein Text (re-)kombiniert und zusammen zu etwas Neuem fusioniert.

Auf formaler Ebene zeigt sich diese Fusion von Textbausteinen insbesondere an einem Buch, dessen Hauptzweck darin besteht, andere Bücher in sich aufzunehmen und durch systematische Bearbeitung deren Inhalte zu etwas Neuem zu verschalten. Es ist kollektives Buch, das seine eigene Abschaffung als Buch vollführt: Ein Buch, das nichts als Bücher verzeichnet, ein Buch, das von vielen Autoren geschrieben ist und noch viel mehr Autoren vereint, weil es deren bibliographische Metadaten listet. Ein Buch, das sich nur in seiner äusserlichen Form noch als Buch tarnt (Abb. ??), obwohl es seine Bestandteile längst schon dissolviert oder herausgelöst hat. Es handelt sich um ein Buch über Bücher, das aus nichts als frei verschiebbaren Zetteln besteht: Ein Katalog, und zwar der erste dauerhafte Zettelkatalog der Bibliotheksgeschichte, der den gesamten Bestand der kaiserlichen Bibliothek verzeichnete. Dass diese Arbeit an einem derart weit verzweigenden Projekt einer dezidierten Arbeitsteilung unterliegt, mag kaum überraschen. Umso konsequenter erscheint die Kodifizierung dieser Tätigkeit, die kaum zufällig schon in derselben Zeit, in der Goethe sein delegierendes Briefdiktatsystem oder die Schwarzen Kabinette ihr Abschreibesystem etablieren, eine algorithmische Struktur annimmt. Die Formierung dieses Verschmelzungs-Algorithmus ereignet sich einmal mehr in Wien um 1780, diesmal allerdings nicht in der Stallburg mit ihrer professionalisierten Postkontrolle, sondern gleich vis-à-vis in der Hofbibliothek.

Während sowohl für mittelalterliche als auch für die überwiegende

^{29.} fetscher:2001.



Abbildung 5: Das Buch der Bücher, nur noch äusserlich als Einheit, innerlich immer schon fragmentiert

Zahl der frühneuzeitlichen Bibliotheken die Aufgabe, ein Verzeichnis der Bestände zu erstellen, in den Hoheitsbereich des zumeist alleinigen Bibliothekars fällt,³⁰ zeigt sich diese individuelle Vorgehensweise im Fall der Bestandsaufnahme der Wiener Hofbibliothek zur Regierungszeit von Joseph II. als unpraktikabel. Infolge von politischen Maßnahmen wie etwa der Auflösung der Jesuitenklöster samt ihrer Bibliotheken findet sich die kaiserliche Bibliothek in Wien unversehens mit einer Welle von Neuzugängen an Büchern konfrontiert, die nicht nur den Bestand innert kürzester Zeit anschwellen lassen, sondern auch die organisatorischen Maßnahmen in Frage stellen, wie neue Bücher erschlossen und in den ebensowenig vollständig bekannten Altbestand eingepasst werden sollen.

Angesichts der schieren Menge nicht nur an neuen Büchern setzt der Vorsteher der Bibliothek, Präfekt Gottfried van Swieten, auf ein minutiöses, arbeitsteiliges Verfahren, eine Instruktion, nach deren Anweisungen endlich alle Bücher der Hofbibliothek vollständig erfasst, beschrieben und eingeordnet werden sollen. Schriftliche Instruktionen zur Katalogisierung sind bis zum Ende des 18. Jahrhunderts keineswegs gängig. Erfolgt die Bestandsbeschreibung doch üblicherweise nur unter der Aufsicht eines Bibliothekars, der die Skriptoren und Hilfskräfte mündlich instruiert, um sodann Ablauf und Ergebnis zu kontrollieren. Van Swietens »Vorschrift worauf die Abschreibung aller Bücher der k.k. Hofbibliothek gemacht werden solle« umfaßt dagegen sieben Punkte, was bei der Beschreibung der Bücher zu beachten sei, ergänzt um einen Ablaufplan sowie eine ausführliche

^{30.} schreiber:1927.

und jedwede Eventualitäten berücksichtigende Verhaltensvorschrift, die – und das ist besonders bemerkenswert – einer *if-then-*Struktur, mit anderen Worten: einer algorithmischen Logik, folgt:

```
I. Muß [...] Sollte [...] so müßte [...]
II. Findet er [...] müssen [...]
III. Findet sich [...] so muß [...]
IV. Wenn man [...] so ist [...]
:
```

Dank dieses derart kodifizierten Plans gelingt es den exekutiven Kräften, namentlich den Skriptoren und Bibliotheksdienern, zwischen dem Frühjahr 1780 und ?? nunmehr alle Texte der Bibliothek auf rund 300'000 Zetteln zu beschreiben, die Angaben zu standardisieren und die Papierträger schliesslich in extra dazu angefertigten Katalogkapseln zu bündeln (Abb. ??). Ausgehend von diesen losen und doch geordneten Stapeln, dem sog. >Josephinischen Katalog<, lassen sich neue Ordnungen anfertigen, seien dies alphabetische, systematische oder chronologische. Die lose in Buchform eingefassten Textfragmente sollen auf diese Weise wieder in echte Buchform überführt werden.

Aus der Parallelität individueller Zeiten eines jeden Buchs werden deren auswuchernde Zweige und Gabelungen wieder zusammengeführt in einem Text über Texte und verschmolzen zu neuer Form, aus deren Ordnung wiederum neues Wissen entsteht. Der *merge*-Algorithmus generiert aus den vielen Büchern einer Bibliothek ein einziges: ihren eigenen Katalog, mit dem viele Autoren zu einer Struktur zusammengebunden werden, die wiederum neue Autoren aufzugreifen und zu produzieren vermag. Aus der Synthese der informationellen Vereinzelungen von Einträgen, die ihrerseits einen immer schon fragmentierten Text repräsentieren wird wieder ein einziger Strang produziert, ein Faden oder Pfad, der mit seinen volatilen Elementen seinerseits und jederzeit neue Verzweigungen oder auch Weiterführungen auf demselben Weg erlaubt.

Coda: Auto(r)Korrektur

Verzweigen, Kopieren und Verschmelzen zählen zu den eminenten Funktionen verteilter Code-Autorschaft in der Softwareentwicklung. Wie zu zeigen versucht wurde basieren diese Funktionen auf Praktiken, die sich zum einen in der Literaturgeschichte und ihrem Zusammenspiel aus experimenteller Autorschaft und Editionstätigkeit gründen (Borges und *branch*), zum zweiten in der Geschichte der Telekommunikation und im Postverkehr (Schwarze Kabinette, *copy*, *diff*) und schließlich auch in den Verwaltungspraktiken des Wissens, konkret in der Katalogarbeit der Aufklärung (Wiener

^{31.} bartsch:1780.

Hofbibliothek und ihre Zettelkatalog, *merge*). Diese historische Konstellierung mag – wenngleich nur schlaglichtartig oder exemplarisch – vor Augen führen, aus welchen Bestandteilen die informatische Versionsverwaltung verfertigt ist, ohne es zu wissen. Ein heuristisches Verfahren, das versucht die beiden Wissensstränge miteinander zu verbinden – also die medialen Praktiken der Code-Entwicklung einerseits mit der historiographischen Quellenkritik andererseits –,³² sollte jedoch nicht allein darauf abzielen, die aktuelle Funktionsweise der *computer science* historisch nach hinten zu verlängern, sondern ebenso umgekehrt, aus und vor allem *in der Tiefe der Geschichte* nach Denkfiguren und Funktionsweisen, nach medialen Praktiken und historischen Tätigkeiten, nach Fragestellungen und Problemlösungen zu suchen, um diese Erkenntnisse in die Softwareentwicklung hineinzutragen. Wie könnte das aussehen?

Die Reichweite eines Befehls wie diff mag für informatische Zwecke begrenzt und aus historisch-kritischer Perspektive für einen Editor zudem keineswegs neu sein; was aber wäre, wenn es einen Befehl gäbe, der Differenzen auch auf einer inhaltlichen Ebene vorschlagen könnte? Also einen Befehl, der sich an Autoren im Schreibprozeß richtet und sich dabei weniger an Herausgeber-Funktionen orientierte als an den Leistungen eines kritischen Lesers oder Lektors, der einem um den guten Ausdruck bemühten Autor mehr anbietet als lediglich diff-gleich nur Abweichungen zu markieren? Was wäre, wenn der Algorithmus, statt bloss die unmerklichen bis infinitesimalen Deltas zu verzeichnen, selbst Kreatives leistete? Wenn beim Schreiben in den Officeprogrammen auf Befehl die Routine eines Lektors abzurufen wäre, der die subtilen Stiländerungen souverän erkennt und seinerseits durch feine Vorschläge variieren kann? Gesucht wäre also so etwas wie ein schlauer Diener beim Schreiben, ein Assistenzsystem für die treffende Formulierung, ein Lektorats-Algorithmus, der nicht nur Goethe und Schiller unterscheiden kann in ihrer jeweiligen Stilistik, sondern auch noch Goethe alternative Vorschläge à la Schiller unterbreitet und umgekehrt, wenn Schiller stockt beim Schreiben, die Vorschläge à la Goethen einspeist.

Ein solcher Algorithmus würde folgende Arbeitsschritte umfassen, indem er einerseits eine informatische Stilanalyse verfolgt, andererseits aber auch eine softwareseitige Stilgenese anbietet, um überhaupt neue Vorschläge unterbreiten zu können. Mit anderen Worten, ein solches System arbeitet mit den Lehren der Vergangenheit, um künftigen Texten den Weg vorzuspuren. Wie sähe die Modellierung eines solchen Stil-Befehls konkret aus?

1. Schritt: Einlesen eines Beispieltexts, eines Text-Korpus, eines ganzen

^{32.} Vgl. zu diesem Ansatz die erste Fassung des vorliegenden Texts, wo statt der Überlegungen zur Autokorrektur die Methode einer »Quellcodekritik« als Verschmelzung von Historiographie und *Literate Programming* entwickelt wird, github.com/nachsommer/VersionsKontrolle/tree/master/1.Fassung.

Werks zur Stilanalyse, also etwa die gesamten Dramen von Shakespeare.

- 2. Schritt: Textanalyse. Da sich diese Art der stilistischen Mimesis oder Originalkopie aus der copia verborum speist, aus der Fülle der Wörter, gilt es die eingelesenen Buchstabenmengen zu gewichten, zu sortieren und zu überprüfen, und zwar mit einer einfachen Statistik auf die Übergangswahrscheinlichkeiten von einem Wort zum nächsten. Mit diesen Reihen von Worten zu Worten zu Worten, die sich ohne grösseren Aufwand computertechnisch etwa mit Hilfe von Markov-Ketten modellieren lassen,³³ erhält jedes Wort in einem Text einen numerischen Wert, der angibt, mit welcher Wahrscheinlichkeit das eine Wort auf ein anderes folgt.
- 3. Schritt: Textsynthese. Nach einer bestimmten Anlernphase, innerhalb derer sich der Algorithmus den Stil eines bestimmten Textes oder gar Autors aneignet, steht ein Repositorium bereit, mit dessen Hilfe beim Formulieren Vorschläge unterbreitet werden können: Ein im Schreibprogramm eingegebenes Wort oder eine Formulierung klingt noch nicht gut genug? Auf Knopfdruck werden Alternativen angeboten, und zwar nicht einfach durch einen Griff in das Synonymwörterlexikon, sondern durch einen Vergleich mit den zuvor gewonnenen Übergangswahrscheinlichkeiten einzelner Passagen, die wiederum stilistisch charakterisiert wurden. Darf's noch etwas Kafka sein?

Was also ein solcher Algorithmus auf Basis einer Markov-Ketten-Analyse oder mit einem DeepLearning-Algorithmus wie word2vec³⁴ liefert, ist eine stilistische Anverwandlung einer bestimmten Autorschaft. Je nachdem, was ihm eingefüttert wird, ergeben sich charakteristische Formulierungshilfen: Beim Einlesen des *Götz von Berlichingen* gäbe es einen mittelalterlichen Alltagssound. Beim Einlesen von Kafkas *Der Prozeß* gäbe es kristallklare Prosa in parataktischem Satzbau. Und beim Einlesen des Autors von *Sein und Zeit* gäbe es einige nur bedingt hilfreiche, weil selbstbezügliche Formulierungsvorschläge, weil die ganze Welt plötzlich zu >welten

beginnt.

Mit einer solchen Anordnung von >mimetischen Algorithmen könnte es gelingen, einerseits Fragen der *Software Studies* ins 18. Jahrhundert zu tragen, also die >alten Praktiken der Exzerpt-, Fragment- und Informationsschnipsel-Verarbeitung auf den Kontext einer kollektiven Autorschaft der Gegenwart zu beziehen. Aber auch umgekehrt eröffnet sich damit ein Weg, und zwar durch eine Analyse der Instruktionen und Verfahren, mit denen kollaborative Autorschaft in den unterschiedlichsten Formen

^{33.} hilgers:2007. 34. Vgl. etwa die Implementierung in Java unter github.com/deeplearning4j/deeplearning4j bzw. demnächst auch in github.com/nachsommer/synapsen.

und Situationen historisch zur Ausführung gelangten, um damit die komplexen, distribuierten, wolkenbasierten Verfahren verteilter Autorschaft der Gegenwart gegenzulesen, das heisst, nicht nur zu verstehen, sondern – historia est magistra codicum – diese Algorithmen selbst aus der Tiefe der Geschichte heraus weiter zu entwickeln, um so zu einer wechselseitigen Erhellung von Code-Entwicklung und historischer Forschung zu gelangen. Doch dazu ist es notwendig, noch andere lose Enden aufzunehmen.

