Hannes Bajohr

Pathos, Vektoren, Nebel

68–74

Sind Programmiersprachen, sind Codes für emotionale und affektive Aspekte offen? Kann man, zugespitzt formuliert, Pathos programmieren? Ich versuche, mich dieser Frage über zwei Seiten zu nähern, 1. über Pathos und Programmierung und 2. über Pathos und KI

1. PATHOS UND PROGRAMMIERUNG

Das erste Problem: Code und Programmiersprachen sind *keine Sprachen*. Man kann nicht Deutsch *und* C++ sprechen wie man Deutsch und Französisch spricht – man kann überhaupt C++ nicht *sprechen*.

Programmiersprachen sind Kalküle zur Verarbeitung von Information. Oder, nach einem alten Satz der Informatik: »Programme = Algorithmen + Datenstrukturen.«[1] Das bedeutet, es gibt eine grundsätzliche Bifurkation von digitaler Sprachlichkeit: einmal die nicht-natürlichsprachigen Regeln zur Datenverarbeitung (der Algorithmus) und dann die Datenstrukturen, die natürlichsprachig sein *können* – z.B. ein als Textdatei codiertes Gedicht –, es aber nicht sein müssen; von Datenbank über Bilddatei oder Vektorgrafik ist alles möglich. Beide, Algorithmen und Datenstrukturen, sind nicht nur separat *gespeichert,* sondern auch – obwohl alphanumerisch codiert – separate *Sinn*komplexe.

Algorithmen sind imperativisch, anweisungshaft; sie haben zunächst keinerlei konnotative, referenzielle oder pragmatische Dimension. Sie treiben damit die Konventionalität von Sprachen ins Extrem: Alle Ähnlichkeiten zu natürlichsprachigen Wörtern sind in Programmiersprachen reine Bequemlichkeit zum Erlernen für menschliche User. Damit sind sie zwar auf *Lesbarkeit* ausgelegt, die ist aber von der Regel- und Anweisungshaftigkeit ihrer Ausführung abgekoppelt. Der Programmierpapst Donald Knuth soll daher gesagt haben: »Programmieren ist die Kunst, einem anderen Menschen zu sagen, was der Computer tun soll.«[2] Alles, was über Verknüpfung dieser menschenlesbaren Anweisungen mit Auswirkungen im Prozessor und Speicher hinausgeht, ist für ihre *Funktion* als Programmiersprache irrelevant. Programmiersprachen sind, wie der Philosoph Charles Taylor es formuliert, »Darstellung ohne Ausdruck«; alle konnotativ-expressive Dimension geht ihnen ab.[3]

Anlass zu diesem Text war der Workshop »Pathos übersetzen«, der am 29./30. März 2022 am LCB stattfand.

1 Nikolaus Wirth, *Algorithms + Data Structures = Programs*, Upper Saddle River, NJ: Prentice-Hall 1976.

2 Ein möglicherweise apokrypher Aphorismus, der seit Jahren durchs Internet geistert, für den ich aber keine Quelle finden konnte. Zitiert etwa in: Selma Takir, »Reading CS Classics«, in: *Communications of the ACM 55,* Nr, 4, S. 32–34, hier S. 33.

3 Charles Taylor, »Theories of Meaning«, in: ders., *Human Agency and Language. Philosophical Papers I,* Cambridge: Cambridge University Press 1985, S. 247–292, hier S. 267.

Die strenggenommen einzigen Zeilen sind also gerade jene, die dem Computer nichts bedeuten.

Wenn es also programmiertes Pathos gibt oder geben kann, dann wohl nur aufseiten der Datenstruktur, nicht des Algorithmus. Das ist aber trivial – die *Verarbeitung* eines Textes wäre dann vom Pathosgehalt dieses Textes völlig getrennt, weil sie, als Verarbeitung, zunächst kein Interesse am natürlichsprachigen und sinntragenden Wert der Datenstruktur hätte und, wie gesagt, statt eines Gedichts auch ein Spreadsheet auslesen könnte.

Diese etwas strenge Position setzt aber voraus, dass man den Code *nur* als Code, und d.h. nur in seiner Funktionalität *als* Code liest oder lesen darf – und dann alles ignoriert, was in der Ausführung keine Rolle spielt – also zum Beispiel die Tatsache, dass bestimmte Kommandos in Programmiersprachen natürlich Konnotationen haben können: »kill« etwa, das einen Prozess beendet; »sleep« für eine Pause im Rechenzyklus, etc. Oder die Tatsache, dass auch Variablen, die Programmierende frei wählen, natürlich etwas bedeuten.[4]

Aber man muss Code nicht *als* Code, also nur in seiner Funktionalität lesen; das macht zum Beispiel die *Code Poetry,* die mit diesem konnotativen Surplus des eigentlich funktional »bedeutungslosen« Codes spielt. Oder die gerade die Differenz zwischen Menschen- und Maschinenlesbarkeit unterläuft, indem sie beide Textsorten mischt, ohne ihre Funktionalität aufzuheben. Letzteres geschieht v.a. in *Kommentaren.* Kommentare sind Zeilen im Code, die nicht ausgeführt werden; sie sind nicht an die Maschine, sondern an die menschlichen Akteure gerichtet – an die Schreibenden selbst oder zukünftige Lesende.

So auch im Gedicht *Drei letzte Worte* der amerikanischen Dichterin Lillian-Yvonne Bertram, das in ihrem 2019 erschienen Lyrikband *Travesty Generator* enthalten ist, den ich als *Farcen-Generator* übersetzt habe. Im Buch ist der Titel des Gedichts stilisiert ist als: »#/usr/bin/python/drei_letzte_worte«. Das sieht so aus wie ein Standardbeginn eines Skrips in der Programmiersprache Python – eine Zeile, die den Ordner angibt, in dem die notwendigen Dateien zur Ausführung liegen (normalerweise »#/usr/bin/python«).

Das Gedicht selbst ist ein Python-Code. Auf der ersten Seite wird eine Permutationsfunktion definiert, die einen Eingabestring (einen Text) durcheinanderwirbelt und wieder an sich selbst anhängt; die Ausführung dieser Funktion auf den folgenden Seiten.

Die Funktion heißt »permutationen«. Sie verarbeitet eine Eingabe, die hier »elemente« heißt; das ist nur die abstrakte Bestimmung der Funktion – hier wird noch nichts ausgegeben, sondern nur die Regel eingeführt, nach der etwas verarbeitet werden kann.

4 Daraus machen die »Critical Code Studies« ein ganzes Forschungsprogramm, indem sie die kulturellen Konnotationen von Code untersuchen, siehe Mark C. Marino, Critical Code Studies, Cambridge, MA: MIT Press 2020.

Der Text auf dieser Seite – Code als reine Denotation – ist aber unterbrochen durch Kommentare. Diese beginnen mit »#« und werden für den Compiler als nicht auszuführen markiert; er ignoriert sie schlicht, als seien sie nicht da. Hier drängt sich also natürlichsprachiger Text als *Annotation* in die ausführbaren Skriptzeilen hinein.

Die strenggenommen einzigen sinntragenden Zeilen – wenn man Code jede expressive Funktion abstreitet – sind also gerade jene, die dem Computer nichts bedeuten. Pathos – wenn es das denn ist – läge hier *ganz jenseits* des Programms; und zwar grundsätzlich.

Das ist aber vielleicht zu einfach; es ging ja die um Frage der *Programmierbarkeit* von Pathos. Pathos wird in diesem Code nicht nur durch den Kommentar, sondern auch durch das Programm vermittelt – und zwar nicht in seinem Ergebnis, sondern in der Art der *Verarbeitung,* d.h. in der Logik, die das Programm zur Anwendung bringt, und die auf der Architektur des Computers beruht.

Dazu muss man sich den Rest des Gedichts ansehen: Auf der nächsten Seite wird die definierte Funktion erstmals aufgerufen: Statt »elemente«, was in der Definition nur ein Platzhalter war, wird nun wirklich eine Zeichenkette verarbeitet: »Ich«. Da die maximale Anzahl an Permutation von »Ich« 6 beträgt, werden die drei Buchstaben entsprechend durcheinandergewirbelt.[5] Es folgt »nicht«, das, da fünf Elemente lang, bereits 5! = 120 Ausgaben generiert. Schließlich: Bei »Ich kann nicht atmen« sind es (mit Leerzeichen) schon 21! = 6.227.020.800 Ausgaben – so viele, dass das Programm crasht und bezeichnenderweise einen »Memory Error« auswirft.

#/usr/bin/python/drei_letzte_worte ist Eric Garner gewidmet, der am 17. Juni 2014 im Würgegriff eines New Yorker Polizisten starb; seine letzten Worte lauteten »I can't breathe«. Die Kommentare, die Bertram in den Code einfügt, beziehen sich wiederum auf Freddie Gray, der in Polizeigewahrsam starb, als er wegen Verdachts auf das Mitführen eines illegalen Messers angehalten wurde.

Der *Memory Error,* die Unmöglichkeit, sich das durch Polizeigewalt hervorgerufene Schwarze Leid zu begreifen, es zu »verarbeiten« und es angemessen zu erinnern, zu memorialisieren – das wäre hier das Affektive des Codes, eine Art performatives Code-Pathos, das allein auf der operationellen Ebene bleibt und überhaupt *nur* durch die denotative Funktion eines mit Symbolen handelnden Systems und ihren Operationsgrenzen zustande kommt. Wie Zach Whalen in seiner detaillierten Besprechung des Gedichts schreibt: »Bertrams Software präsentiert die Flüchtigkeit der Erinnerung als Verschränkung von poetischer Darstellung, polizeilicher Brutalität und technischer Limitierung. Indem sie das Skript zum Scheitern verurteilt, lädt

5 Im englischen Original steht hier »I«, was natürlich überhaupt nicht permutiert werden kann – sich aber leider in der Übersetzung verliert.

Bertram uns ein, das Trauma dieser Brutalität nachzuempfinden und ihre technischen und politischen Folgen zu interpretieren.«[6]

Was hier scheitert, ist nicht Begreifen oder Verstehen oder Sinn aufseiten des Systems; dort versagt die bloße Verarbeitung von Symbolen, jenseits von ihrer Bedeutung *für uns.* Alles, was an Sinn hier wieder hineinkommt, steht bereits außerhalb des Systems und ist Folge der Rahmung, der Inszenierung, die Bertram aufführt und die wir erst interpretieren müssen. Der Philosoph Maurizio Lazzarato meint genau diesen Umstand, wenn er schreibt: »Zeichenmaschinen (Computer) arbeiten vor und neben Bedeutung – sie produzieren einen ›Sinn ohne Bedeutung‹, einen ›operativen Sinn‹.«[7] So ein operativer Sinn scheint mir bei Bertram meisterhaft zum Zug zu kommen.

2. PATHOS UND KI

Code *an sich* ist scheint jedenfalls kaum geeignet, Pathos, Affekt, Emotion zu produzieren. Bei Bertram lag die Pathosfunktion in der Einbettung in einen interpretativen Kontext und in einer Art externer Metareflexion auf die Logik der Maschine. Gibt es aber Arten der Verarbeitung, die selbst eine Pathosfunktion haben? Ich glaube, in *klassischen* Programmiersprachen (wie etwa Python) ist das nicht der Fall. Hier gibt es eher eine Art notwendiger Kälte – wie eben in der Permutation, der Kombination aller Elemente eines Korpus. Gut möglich, dass auch Wiederholung Pathos erstehen zu lassen vermag, dafür fällt mir aber kein Beispiel ein. Viel einfacher dagegen sind Absurdität und Witz zu generieren: Die Inkongruenz, die für sie ausschlaggebend ist, hat man schnell per Rekombination hergestellt. (Wie im Kinderspiel »MadLibs«, das aus mit frei gewählten Wörtern zu füllenden Schablonensätzen besteht.) Pathos ist vielleicht auch eine Art Inkongruenz – nämlich zwischen einem Maß und seinem Zuviel –, aber wohl eine, die weitere, umfassendere Sinnstrukturen benötigt, die im Hintergrund mitlaufen können müssen, und die die Grenze festlegen, an der es zum pathetischen Zuviel kommt.Pathos ohne Referenz wäre das Einzige, was ein Computer

Pathos ohne Referenz wäre das Einzige, was ein Computer herstellen könnte; und das Fehlen von Referenz ist womöglich eher komisch als erhaben.

6 Zach Whalen, »Code Critique / Book Review: Travesty Generator by Lillian-Yvonne Bertram«, 1. Februar 2020, http://wg20.criticalcodestudies.com/index.php?p=/discussion/99/code-critique-book-review-travesty-generator-by-lillian-yvonne-bertram.

7 Maurizio Lazzarato, *Signs and Machines. Capitalism and the Production of Subjectivity,* Cambridge, MA: MIT Press 2014, S. 24.

herstellen könnte; und das Fehlgehen von Referenz ist womöglich eher komisch als erhaben.

Vielleicht sind maschinelles Lernen oder KI hier Möglichkeiten; denn sie bieten eine grundsätzlich *andere* Art, mit digitaler Sprache umzugehen, als der klassische Code, wie er bei Bertram verwendet wird.[8] ML wird oft abschätzig als »statistics on steroids« bezeichnet, als sehr komplexe Form stochastischer Berechnung von Sprache, also eben nur Berechnung. Wichtig ist dabei aber zweierlei:

Erstens sind *statistische Verteilungen,* die ein Sprachmodell produziert (welche Buchstaben oder Token mit welchen am ehesten korrelieren), nicht mehr dasselbe wie Algorithmen: Ich kann das Modell prinzipiell nicht mehr in einzelne, klar zu verfolgende Regelschritte zurückübersetzen. Das meint nicht allein die »Intransparenz«, die Black Box der KI, dass man also nicht mehr genau weiß, *wie* ein Ergebnis zustande gekommen ist, sondern schlicht, dass sie *anders* funktionieren – das heißt, dass die Begrenzung von klassischem Code womöglich *nicht* auch die Beschränkung von KI-Modellen ist.

Zweitens ist wichtig: Zwar ist Sprache auch in der KI nur Material, das verarbeitet wird; aber die *Kodierung* dieser Sprache funktioniert ganz anders; nämlich als vieldimensionale Vektoren, d.h. komplexe Listen von Zahlen, die das Verhältnis zu allen anderen Wortvektoren immer mitbeinhalten.

Sprache ist hier tatsächlich, wie bei Saussures Strukturalismus oder Quines semantischem Holismus, allein durch ihr *Verhältnis zu den anderen Elementen der Sprache* definiert; und weil hier nicht nur Verhältnisse von Wörtern zu *Wörtern,* sondern auch Verhältnisse *von Verhältnissen* codiert werden – Verhältnisse zweiten Grades –, können diese Vektormodelle auch jene Relationen einbeziehen, die in bloßen Regeln nicht oder nur schwer zu fassen sind.

Ein Beispiel ist dabei immer die Implikation von Kategorien, die auf Wortebene gar nicht codiert ist, etwa *Gender*. Ein berühmtes Beispiel von Vektorenarithmetik lautet: »king – man + woman = queen«.[9] Gender ist hier ein bloßes Ergebnis von komplexen Relationen von Vektoren. (Das ist vor allem als ungewollter *bias* problematisch – so dass die Ausgaben einer KI Rassismus reproduzieren können, ohne dass ihre Trainingsdaten ausdrücklich rassistisch sein müssen.) Wenn Sprachmodelle *implizites* Wissen *explizit* machen können, und eben so etwas wie soziales Geschlecht zur Erscheinung bringen, warum nicht auch andere soziale Dimensionen von Sprache – und

8 Zum Unterschied zwischen klassischer Programmierung und Machine Learning – oder sequenziellem und konnektionistischem Paradigma –, siehe Hannes Bajohr, »Algorithmische Einfühlung. Für eine Kritik ästhetischer KI«, in: ders., *Schreibenlassen. Texte zur Literatur im Digitalen,* Berlin: August 2022, S. 131-172

9 Siehe Tomáš Mikolov, Wen-Tau Yih und Geoffrey Zweig, »Linguistic Regularities in Continuous Space Word Representations«, in: Proceedings of the 2013 Conference of the NAACL, Atlanta, Georgia: ACL 2013. Das entwickle ich ausführlich in: Hannes Bajohr, »Dumme Bedeutung. Künstliche Intelligenz und artifizielle Semantik«, in: *Merkur,* November 2022.

also Pathos? Ich sehe keinen unmittelbaren Grund, warum das nicht möglich sein sollte.

Natürlich haben auch Sprachmodelle kein Weltwissen, können kein Pathos »fühlen«. Es ist klar, dass ihnen – außer Syntax – sonst so gut wie alles fehlt, was zu Sprache gehört – Semantik und Pragmatik, Weltreferenz und das vorsprachliche *tacit knowledge,* das jedem sinnerschließenden Weltumgang zugrunde liegt. Daher darf man sich von großen Sprachmodellen wie GPT-3 und den erstaunlich »natürlichen« Ausgaben, die sie produzieren, nicht täuschen lassen: *Verstehen, Sinn* und so weiter, gibt es auch hier immer nur auf Userseite – als Illusion oder Projektion.

Verabschiedet man sich aber von Sinn als intentionaler Kategorie, die den Willen zur Kommunikation, zu einem Weltmodell oder einer Vorstellung vom Bewusstsein anderer hat, und denkt sie allein als Verhältnis von Verhältnissen innerhalb eines *Vektorraums von Sprache* – in denen damit auch nicht direkt codierte Begriffe, Stile, Stimmungen, »vibes«[10] zum Vorschein kommen können –, dann sehe ich nicht, wieso sich so etwas wie Pathos nicht genauso extrahieren ließe wie Gender.

Sinn wäre dann ein *Interferenzeffekt* von Daten, als etwas, das sich einfach ergibt oder das zutage tritt ohne ein dahinterstehendes Bewusstsein. Sprache – und Pathos – wäre hier dann materialistisch verbrämt. Lydia Liu hat dafür die schöne Metapher des Telefonsystems, dem ja auch egal ist, was durch seine Kabel läuft. Sie schreibt: »Die materielle Sprache [...] funktioniert wie eine Telefonvermittlungsstelle oder eine kybernetische Maschine, die automatisch läuft, unabhängig davon, was in ihren Leitungen passiert.«[11] So wäre dann auch Pathos per KI eben nur »dumm«,[12] hinter dem Rücken der Maschine zu haben.

Pathos wäre gerade *nicht* »programmierbar«, sondern nur »extrahierbar«. Es würde sich als komplexe statistische Abhängigkeit aus genügend umfangreichen Korpora so erheben, wie sich etwa Nebel über dem Verwesungsprozess eines Moors, als Ausdünstung toter Materie erhebt.

So wäre dann auch Pathos per KI eben nur »dumm«, hinter dem Rücken der Maschine zu haben.

10 Peli Grietzer, »A Theory of Vibe«, in: Glass Bead, Nr. 1 (2017), http://www.glass-bead.org/article/a-theory-of-vibe.

11 Lydia H. Liu, The Freudian Robot. Digital Media and the Future of the Unconscious, Chicago: University of Chicago Press 2010, S. 189f.

12 Siehe Bajohr, »Dumme Bedeutung«.

KING — MAN

+

WOMAN = QUEEN