Heuristische Re-Digitalisierung von deutschsprachigen Diskettenmagazinen

Anton Ehrmanntraut

31. März 2022

1. Einleitung

Die vorliegende Arbeit beschäftigt sich mit *Diskmags* – Fachmagazine, welche in den 1980er und 1990er Jahren publiziert wurden, und dabei über Disketten als digitale Datenträger vertrieben wurden. Diese wurden zu großem Teil für den 8-Bit-Heimcomputer *Commodore 64* (C64) als Wiedergabegerät konzipiert: Konsumierende konnten mit ihrem jeweiligen C64 die Diskmag-Disketten abspielen, und – ähnlich zu modernen Webseiten – durch die Inhalte der Diskmags ›browsen‹. Neben Text-Inhalten, die unter anderem Rezensionen aktueller Spiele, Diskussionen über Hardware, Editorials, Leserbeiträge umfassten,[1] waren auch interaktive Komponenten wie Computerspiele, Programme, und Demos auf den Diskmags vorhanden, welche von den Konsumierenden direkt gestartet werden konnten. Ferner, hierauf machen Roeder und Rettinghaus aufmerksam, war die Präsentationsform der Diskmags spezifisch multimedial und interaktiv: hier sind beispielsweise die mit Animationen gepaarte Menüführung oder die computergenerierte Hintergrundmusik zu nennen.[2]

An dieser Stelle muss hervorgehoben werden, in welchem Zustand sich die akademische Bearbeitung mit diesen kulturellen Artefakten befindet. Roeder und Rettinghaus stellen fest: »Die Überlieferungssituation der Diskmags ist prekär.«[3] Die Datenträger sind in beinahe keinen öffentlichen Beständen aufgenommen. Die 8-Bit-Szene hat zumindest von einigen Diskmags digitale Abbilder angefertigt, sodass diese online verfügbar sind und bspw. von C64-Emulatoren eingelesen werden können. (Diese Abbilder sind elektronische Repräsentationen der Disketten, wie sie ein Diskettenlaufwerk sehen würde.) Bei den vielen noch nicht gesichteten und noch nicht gesicherten physikalischen Datenträgern besteht Gefahr, dass sie permanent unlesbar werden. Neben der Archivierung ist auch eine ernsthafte textkritische Aufarbeitung der Diskmags kaum vorhanden. In diesem Zusammenhang ist auch die Szene um Diskmags als kulturelle Gemeinschaft und deren immaterielle Kultur kaum dokumentiert; Die wenigen überlieferten Informationen liegen meist nur in Form von grauer Literatur vor.[4] Es ist wenig bekannt und gesichert über die Akteurinnen

1. Torsten Roeder und Klaus Rettinghaus, »Game On! Digitale Archäologie und Edition zu(m) Spielen«, in *Abstract zur Konferenz Digital Humanities im deutschsprachigen Raum 2020*, DHd 2020. Spielräume: Digital Humanities zwischen Modellierung und Interpretation. 7. Tagung des Verbands »Digital Humanities im deutschsprachigen Raum«, 5. März 2020 (Paderborn, 22. Februar 2020), 138, https://doi.org/10.5281/zenodo.3666690.

2. Roeder und Rettinghaus, 140.

3. Roeder und Rettinghaus, 139.

4. Vgl. Wikipedia: Die freie Enzyklopädie, s.v. »Diskmag«, zuletzt geändert am 28. November 2021. Besucht am 26. März 2022, https://de.wikipedia.org/w/index.php?title=Diskmag&oldid=217678518; vgl. C64-Wiki, s.v. »Diskettenmagazin«, zuletzt geändert am 3. Januar 2022, besucht am 26. März 2022, https://www.c64-wiki.de/index.php?title=Diskettenmagazin&oldid=247552

und Akteure, das Ausmaß dieser Subkultur, deren Praktiken (Wie programmiert und vertreibt man Diskettenmagazine? Wie verlief die Kommunikation an die Verlage zurück?) und Diskurse (Wie werden z.B. Computerspiele oder die Ästhetik von Diskmags bewertet?), innerhalb dieser lose zusammenhängenden Community, die als eine der ersten über digitale Wege (noch vor dem Internet) kommuniziert haben muss.[5]

Mit diesem Hintergrund sprechen sich Roeder und Rettinghaus dafür aus, auch die digitalen Diskmags wissenschaftlich und kritisch zu kommentieren, gerade wegen deren durchaus hohen kulturgeschichtlichen Relevanz. Sie regen an, auch für Diskmags Editionen zu erarbeiten, wie es bei den analogen kulturellen Kulturerben üblich ist.[6] Eine digitale Edition auf Basis einer TEI-Codierung wurde beispielhaft von Roeder für die Diskmag *Magic Disk 64*, Ausgabe 1987/11, angefertigt.[7] Während die Methodik einer Edition schon an sich mit Schwierigkeiten des Mediums umgehen muss (insbesondere die Frage, wie bzw. ob Nutzererfahrungen wie die oben genannte Interaktivität und Multimedialität, aber auch Erfahrungen wie die langen Ladezeiten oder die 8-Bit-Grafik abgebildet werden soll), ist ein essenzieller Arbeitsschritt die Übertragung des Textinhalts des Diskmag. Gegenwärtige Computersysteme können den auf den Diskmags gespeicherten Textinhalt nicht trivial lesen und extrahieren. Das liegt zum einen an den Weiterentwicklungen von Dateisystemen und Textcodierungen, zum anderen auch daran, dass zur Entwicklungszeit der Diskmags keine festen Normierungen von Dateien existierten. Ohne technisches Spezialwissen bleibt Editorinnen und Editoren nur, den Text vom C64-Bildschirm abzutippen oder durch optische Zeichenerkennung (OCR) Bild für Bild auszulesen.

Beitrag: Aus dieser Schwierigkeit heraus entwickelt diese Arbeit ein Python-Programm[8], um eine Textextraktion aus den digitalen Abbildern von Diskmags so automatisiert wie möglich und mit nur wenig Spezialwissen durchzuführen (*Diskmag-Textextraktion*). Im Speziellen soll die Software helfen, die Inhalte der Zeitschriftenseiten von Diskmags zu extrahieren (also z.B. nicht die in den Computerspielen vorkommenden Texte). Das soll zum einen der Entwicklung von Diskmag-Editionen beitragen: mit dem Programm kann die Textübertragung übersprungen werden, und unmittelbar mit der kritischen Auseinandersetzung des Texts begonnen werden. Zum anderen zielt diese Arbeit und das Programm auch auf einen *distant reading*-Einsatz: anstatt jede einzelne Diskmag-Ausgabe sorgfältig qualitativ zu analysieren, ermöglichen quantitative maschinelle Methoden auf großen Korpora von Diskmag-Texten das Aufdecken von stilistischen und zum Teil auch inhaltlichen Regularitäten unter diesen Texten (bspw. stilometrisches Clustering, *topic modeling*, usw.).[9] Mit diesem Hintergrund bezieht die hier vorliegende Arbeit das deutschsprachige Diskmag-Archiv *c64.at*[10] (im folgenden *C64-Archiv*) ein, und versucht möglichst viele deutschsprachige Inhalte aus den Abbildern zu extrahieren. Zusätzlich werden im Folgenden Einsichten bezüglich dem Aufbau dieses Archivs präsentiert, welche sich aus der Auswertung der Programmergebnisse zeigen.

Folgender Abschnitt 2 geht auf die Schwierigkeiten der Diskmag-Textextraktion ein, und präsentiert Details der Implementation des oben bereits angesprochenen Python-Programms. Abschnitt 3 evaluiert mit Stichproben die Güte der implementierten Heuristik. Anschließend wird das Programm in Abschnitt 4 auf dem oben genannten C64-Archiv angewendet. Letzter Abschnitt 5 fasst die Beobachtungen und Erkenntnisse über das Programm und das Diskmag-Archiv zusammen.

5. Roeder und Rettinghaus, »Game On! Digitale Archäologie und Edition zu(m) Spielen«, vgl.

6. Vgl. Roeder und Rettinghaus.

7. Torsten Roeder, »Magic Disk 64 – November 1987«, zuletzt geändert am 26. Februrar 2020, besucht am 19. März 2022, http://diskmags.elitepiraten.de/md_87-11.html.

8. https://github.com/aehrm/diskmag-hausarbeit-2022. Das Programm ist auch im Anhang der digitalen PDF dieses Dokuments verfügbar.

9. Vgl. z.B. Fotis Jannidis, Hubertus Kohle und Malte Rehbein, Hrsg., *Digital Humanities: eine Einführung* (Stuttgart: J.B. Metzler Verlag, 2017), Kap. 20, ISBN: 978-3-476-02622-4, https://doi.org/10.1007/978-3-476-05446-3.

10. Ferdinand Gansberger, »www.c64.at. Das Verzeichnis von deutschsprachigen Magazinen für den C64«, besucht am 20. Februar 2022, https://c64.at/.

2. Implementierung

Die Extraktion der Dateien an sich aus den (Abbildern der) Diskmags stellt keine Hürde dar. Das von den Commodore-Diskettenlaufwerken benutzte CBM-Dateisystem organisiert die Diskette in Dateien. Die zugehörige Schnittstelle versteht Dateien als Sequenz fester Länge von Bytes, also wie gegenwärtige Dateisysteme. Die technischen Details betreffend den C64 und auch das CBM-Dateisystem sind – im Kontrast zur dünn vorliegenden kulturellen Dokumentation – sehr gründlich detailliert, da zur Hardware noch Bedienungsanleitungen mitgeliefert wurden, die umfassend die Funktionsweise dokumentieren. So ist beispielsweise im *Commodore 1541 Disk Drive User's Guide*[11] des Diskettenlaufwerks eine vollständige Beschreibung des verwendeten Dateisystems enthalten. Aus diesem Umstand heraus existieren gegenwärtig schon für das Programm relevante Python-Pakete. Eines, welches das Auslesen der Dateien aus einem Diskmag-Abbild ermöglicht,[12] und eines, welches die Verwendung von PETSCII, die für den C64 spezifische Codierung, in Python anbietet.[13] Letzteres baut auf einer PETSCII-nach-Unicode-Abbildung von Linus Walleij auf.[14]

2.1. Klassifikation

Das zentrale Problem liegt in dem Erkennen von (Plain-)Text-Dateien. Insbesondere muss alleine durch den Inhalt einer Datei eine Text-Datei von anderen Dateitypen unterschieden werden, unter anderem BASIC-Programme, Maschinensprache-Programme oder weitere nicht ausführbare Anwendungsdateien wie z.B. Spielstände. Das automatische Erkennen, in welcher Sprache ein gewisses Dokument verfasst wurde (*Language Identifcation*, LI), ist intensiv erforscht und gilt im Wesentlichen als gelöst.[15] Im Allgemeinen werden hier statistische Regularitäten von natürlichen Sprachen ausgenutzt, um ein Dokument einer Sprache zuzuordnen. Hierbei ist aber zu betonen, dass gängige LI-Ansätze grundsätzlich davon ausgehen, dass das fragliche Dokument in überhaupt einer natürlichen Sprache verfasst ist. Methoden der LI, welche auch mit ›Rauschen‹ in den Eingangsdaten umgehen können, sind dagegen noch weitestgehend unerforscht.[16] Damit sind die gängigen Methoden der LI für die hier vorliegende Fragestellung wenig anwendbar: zum einen, weil die primäre Aufgabe darin besteht, Text- von Nicht-Text-Dateien zu trennen. Zum anderen, weil selbst die Text-Dateien durch Artefakte des C64-Systems ›verrauscht‹ sind (vgl. Abschnitt 2.2). Dies ließ sich auch experimentell während der Entwicklung des hier präsentierten Programms beobachten. Auch generelle Methoden des maschinellen Lernens (sowohl überwacht als auch unüberwacht) sind nur schwer anwendbar, da keine geeigneten Trainings- bzw. Test-Datensätze in hinreichender Größe existieren.

Anstelle statistischer Regularitäten der einzelnen natürlichen Sprachen verwendet daher das hier vorliegende Programm zur Klassifikation eine sehr einfache Heuristik auf Basis von (1) Redundanz in Dateien, und (2) Zeichenklassen. (Damit hat das Verfahren auch den Vorteil, dass sie unabhängig der zugrunde liegenden natürlichen Sprache ist, und z.B. auch englischsprachige Diskmags verarbeiten kann.) Eine Datei mit byteweisem Dateiinhalt X klassifiziert das Programm wie folgt:

11. Commodore Business Machines Electronics Ltd., *Commodore 1541 Disk Drive User's Guide* (1982), besucht am 24. Februar 2022, https://archive.org/details/Commodore_1541_Disk_Drive_Users_Guide_1982-09_Commodore.
12. Simon J. Rowe, *d64*, Version 1.6 (28. August 2021), Python-Paket im *Python Package Index*, besucht am 24. Februar 2022, https://pypi.org/project/d64/.
13. Irmen de Jong, *cbmcodecs2*, Version 1.2 (21. Oktober 2021), Python-Paket im *Python Package Index*, besucht am 24. Februar 2022, https://pypi.org/project/cbmcodecs2/.
14. Linus Walleij, »PETSCII to Unicode mapping«, besucht am 20. Februar 2022, http://www.df.lth.se/~triad/krad/recode/petscii.html.
15. Vgl. z.B. den Survey von Tommi Jauhiainen u.a., »Automatic Language Identification in Texts: A Survey«, *Journal of Artificial Intelligence Research* 65 (25. August 2019), https://doi.org/10.1613/jair.1.11675.
16. Vgl. Jauhiainen u.a., 734.

(1) Ist die Entropie $H(X)$ (hier sei je ein Byte als eines von 256 Zeichen zu verstehen) mindestens 7 bit pro Byte, wird die Datei als *komprimiert* klassifiziert.

(2) Liegt die Entropie $H(X)$ unter 7 bit pro Byte, und sind gleichzeitig mindestens 50 % der (PETSCII-codierten) Zeichen in X Buchstaben, wird die Datei als *Textdokument* klassifiziert.

(3) Andernfalls, falls (1) und (2) nicht zutreffen, wird die Datei als *unbekannt* klassifiziert.

Die Analyse der Entropie einer Datei soll ausschließen, dass komprimierte Dateien (ggf. mit vielen Buchstaben) als Textdokumente erkannt werden. Die Kompression von Dateien zum Sparen von Speicherplatz auf den Disketten war übliche Praxis der C64-Nutzenden dieser Zeit, weswegen komprimierte Dateien explizit erkannt werden müssen.[17] Auch wenn die verwendeten Kompressionsalgorithmen unbekannt sind, können wir komprimierte Dateien anhand ihrer Entropie erkennen. Kompression (in jeglicher Form) zielt darauf ab, die Redundanz einer Datei zu minimieren, also die Entropie $H(X)$ auf bis zu 8 bit pro Byte zu maximieren. In diesem Sinne ist eine relativ hohe Entropie ein deutliches Zeichen für das Vorliegen einer komprimierten Datei. Umgekehrt ist bekannt, dass Plain-Text in der Regel eine Entropie um 4 bit pro Byte aufweist.

Die Analyse des relativen Anteils von Buchstaben soll ausschließen, dass insbesondere Programmcodes nicht fälschlicherweise als Texte identifiziert werden. Reine Programmcodes bestehen in der Regel aus nur wenigen Buchstaben. Das gilt sowohl für Maschinensprache-Programme als auch für BASIC-Programme. (Bei abgespeicherten BASIC-Programmen werden die Befehle/Tokens als je ein einzelnes Byte codiert, welche außerhalb des PETSCII-Buchstabenbereichs liegen.) Die Annahme besteht nun darin, dass in ausführbaren Programmdateien in der Regel selten große Texte als Datensegmente hinterlegt sind, sondern diese ab hinreichender Größe in separate Dateien ausgelagert werden.

2.2. Nachbearbeitung

In einem abschließenden Nachbearbeitungsschritt müssen bei klassifizierten Textdokumenten noch zwei Artefakte des C64-Systems korrigiert werden.

Umlaute: Das C64-System bzw. die zugehörige Codierung hat die deutschsprachigen Umlaute nicht definiert. In der Praxis wurden daher Umlaute mit unbenutzten Grafikzeichen ersetzt, und zusätzlich die C64-Schriftart an den entsprechenden Stellen überschrieben, sodass in der grafischen Ausgabe die Umlaute an den passenden Stellen gezeichnet werden (vgl. Abbildung 1).

Das Programm führt ein ›Zurückrechnen‹ dieser Recodierung über die Maximum-Likelihood-Methode durch. Aus möglichen Recodierungen von Ersatzzeichen (im Text vorkommende Grafikzeichen) zu Umlauten wird diejenige ausgewählt, welche am plausibelsten erscheint, im Sinne von höchst möglichster Likelihood. Die Likelihood wird hierbei über die Wahrscheinlichkeit des Auftretens von beteiligten Trigrammen errechnet. Für ein Trigramm *xuy* mit Umlaut *u* und beliebigen Zeichen *x* und *y* schätzen wir die Wahrscheinlichkeit über die Frequenzen aus einem deutschsprachigen Referenzkorpus.[18] Die Likelihood einer Recodierung bezüglich eines Dokuments ist dann das Produkt der Wahrscheinlichkeiten über alle betreffenden Trigramme im Dokument mit Umlaut als mittleres Zeichen.

Das Programm wählt automatisch die Recodierung mit höchster Likelihood aus und wendet diese auf das Dokument an, bevor dieses geschrieben wird.

Zeilenumbrüche: In einigen Diskmags wird der Bildschirminhalt als Text sofort in den Bildschirmspeicher geschrieben. In diesem sehr einfachen Verfahren werden die Zeilenumbrüche nicht mitgeschrieben, denn der Video-Interface-Chip springt selbstständig nach jeweils einer

17. Vgl. C64-Wiki, s.v. »Packer«, zuletzt geändert am 26. Dezember 2017, besucht am 26. März 2022, https://www.c64-wiki.de/index.php?title=Packer&oldid=179893.

18. Der hier verwendete Referenzkorpus baut auf Wikipedia-Artikeln auf.


```
[...]
\rDinge zu ┘berlegen die
\rdazu beitragen das die Aktivit▮t unser-
\rerer Scene gesteigert wird.
\rParties sind immernoch ein Platz wo
\rStuff released wird.
\r
\rObwohl wir manchmal nicht allzustolz
\rauf die Qualit▮t sein k▮nnen, ist es
\rdennoch etwas aufbauendes.
\rWir m┘ssen akzeptieren
[...]
```

Abbildung 1: Ausschnitt aus der Diskmag *Pulp*, Ausgabe 3, April 2000. Links: Screenshot des C64-Bildschirms, emuliert durch VICE.[19]Rechts: Ausschnitt aus der entsprechenden auf dem Abbild gespeicherten Text-Datei. Beob. dass in der Text-Datei ersatzweise Grafikzeichen zur Repräsentation von Umlauten verwendet wurden. Die Diskmag selbst verwendet eine eigene Schrift. Damit werden Kleinbuchstaben als Kapitälchen dargestellt, und einige Grafikzeichen als Umlaute verwendet.

festen Anzahl an Zeichen pro Zeile in die nächste Zeile. In diesen Situationen würde bei der vom Programm extrahierten Dokument regelmäßig Wörter aneinander stehen, welche durch Zeilenvorschübe getrennt sein müssten.

Unter Information der Zeilenlänge lassen sich die Zeilenvorschübe wieder in das Dokument einsetzen. Die Zeilenlänge lässt sich wieder über die Maximum-Likelihood-Methode abschätzen, analog zu obigen Zurückrechnen der Umlaut-Codierung. Für jede Zeilenlänge n erhalten wir mehrere Trigramme am Beginn der Zeilen. Einige dieser Trigramme folgen nicht einem Bindestrich (in der vorigen Zeile); diese müssten dann also ein Wort beginnen. Als Likelihood für die Zeilenlänge n setzen wir daher die Wahrscheinlichkeit für das Auftreten dieser Trigramme als Wortbeginn, wieder abgeschätzt durch Frequenzen aus einem deutschsprachigen Referenzkorpus.

Das Programm ermittelt automatisch die plausibelste Zeilenlänge und setzt entsprechend Zeilenvorschübe bei Dateien, welche als Startadresse den Bildschirmspeicher angeben.

3. Evaluation

In diesem Abschnitt wird das beschriebene Verfahren durch Stichproben evaluiert, um im nächsten Abschnitt eine informierte Auswertung des C64-Archiv durchzuführen. Hierfür schätzen wir zum einen die Güte der Klassifikation in Text- bzw. Nicht-Text-Dateien. Zum anderen wird überprüft, ob die beschriebenen Nachbearbeitungen zu korrekten Ergebnissen kommt. An dieser Stelle muss betont werden, dass sich der Entwicklungsprozess zum wesentlichen Teil am C64-Archiv orientiert hat, und dieses bei der Entwicklung mit einbezogen wurde – der gleiche Datensatz, der hier für die Evaluation herangezogen wird. Es liegt hier also keine strikte Trennung zwischen Entwicklungs-Datensatz und Test-Datensatz vor, und das mögliche Auftreten von Überanpassung (*overfitting*) muss explizit mitgedacht werden. Gleichzeitig ist gerade die beschriebene Klassifikation ein besonders einfaches und allgemeines Verfahren, weswegen die Gefahr gering sein sollte, die Güte des Klassifikators aufgrund von Überanpassung zu überschätzen.

Klassifikation: Zur Überprüfung der Klassifikation wurden 178 Dateien ausgewählt, welche auf den Diskmags des C64-Archivs gespeichert waren. Auf diesen Dateien wurde die Zuordnung der Dateien händisch überprüft. Aus der Kontingenztabelle 1 ergibt sich eine Genauigkeit (*precision*) von 95.8 % (95 %-KI: 87.9 %–100 %), und eine Trefferquote (*recall*) von 71 % (KI: 57 %–87 %).

19. VICE Team, *VICE: The versatile Commodore Emulator*, Version 3.6.1 (2022), besucht am 24. Februar 2022, https://vice-emu.sourceforge.io/

	Klassifikation		
Ist Text-Datei?	Positiv	Negativ	Insgesamt
Ja	23	9	32
Nein	1	145	156
Insgesamt	24	154	178

Tabelle 1: Kontingenztabelle des Klassifikators auf einer Stichprobe des C64-Archivs.

Diese Beobachtungen stimmen mit der Erwartung überein: die Heuristik kann sehr gut erkennen, wenn eine vorliegende Datei *keine* Text-Datei ist; wenn Dateien vom Programm ausgegeben werden, dann sind diese so gut wie immer tatsächlich Textbeiträge. Umgekehrt sehen wir an der relativ niedrigen Trefferquote, dass einige Text-Dateien nicht von der Heuristik als solche erkannt werden. In den meisten der falsch-negativen Fälle der Stichprobe waren zu viele Sonderzeichen in den Dateien enthalten, weswegen diese fälschlicherweise als *unbekannt* klassifiziert wurden.

Unabhängig von der Güte des Klassifikators wurde durch die Stichprobe eine relative Häufigkeit von Text-Dateien im C64-Archiv mit 18 % (KI: 13 %–23 %) beobachtet. Damit hätte ein *no-skill* Klassifikator (»Die vorliegende Datei ist *immer* eine Text-Datei«) ein F1-Score von 30.5 %, während bei dem hier präsentierten Klassifikator ein besserer F1-Score von 81.6 % (KI: 69.2 %–93.0 %) auf der Stichprobe beobachtet wurde. Zusammenfassend lässt sich aus den Beobachtungen ableiten, dass trotz der einfachen Heuristik der Klassifikator im Wesentlichen gut arbeitet. Insbesondere die hohe Genauigkeit erlaubt den Einsatz des Klassifikators zur automatischen Diskmag-Textextraktion beim Aufbau eines Korpus von Diskmag-Texten.

Umlaute: Zur Überprüfung der Nachbearbeitung wurde eine Stichprobe von Dateien aus Diskmags des C64-Archivs gewählt, bei denen das Programm eine automatische Recodierung für Umlaute durchgeführt hat. In 59 von 63 Text-Dateien wurden die Umlaute korrekt zugeordnet, in den verbleibenden vier Dateien wurde je ein Umlaut falsch zugeordnet. Hierbei ist festzuhalten, dass die vier Text-Dateien jeweils sehr kurz waren, und entsprechend nur wenig Datenbasis für die Inferenz der Zuordnung vorhanden war. An dieser Stelle wird auf die Untersuchung von Fällen verzichtet, bei denen eine Recodierung fälschlicherweise *nicht* vorgenommen wurde: Die verwendete Mustersuche nach geeigneten Trigrammen findet alle relevanten Positionen, an denen Grafikzeichen durch Umlaute ersetzt werden müssten.

Zeilenumbrüche: Ausschließlich die Diskmag-Reihe *Game On* machte von der Möglichkeit Gebrauch, den Bildschirminhalt als Text sofort in den Bildschirmspeicher zu schreiben. Obwohl das Programm die eigentliche Zeilenlänge von 53 Zeichen nicht explizit speichert, hat das Programm in beinahe allen Fällen die korrekte Zeilenlänge ableiten können. Nichtsdestotrotz ist hieraus im strikten Sinn keine Aussage zur Qualität der automatischen Erkennung der Zeilenlänge möglich, denn dieselbe Diskmag-Reihe *Game On* und deren Funktionsweise war Bestandteil des Entwicklungsprozesses; eine Überanpassung an das vorliegende Diskmag-Archiv kann nicht ausgeschlossen werden.

4. Analyse des Diskmag-Archivs c64.at

Nachdem die Stärken und Schwächen des präsentierten Programms quantitativ untersucht wurden, wird in diesem Abschnitt das Programm auf sämtliche Diskmag-Abbilder des bereits in der Einleitung vorgestellten Diskmag-Archiv *c64.at*[20] angewendet. Insbesondere soll hier geprüft werden, ob aus dem Archiv durch automatische Methoden große Mengen an Texte zu einer quanti-

20. Gansberger, »www.c64.at. Das Verzeichnis von deutschsprachigen Magazinen für den C64«.

Klassifikation	Anzahl Dateien	Dateigröße	(Anteil Dateigröße)
unbekannt	6624	24.71 MB	49.9 %
komprimiert	2069	21.07 MB	42.5 %
Text-Datei	963	3.73 MB	7.5 %
insgesamt	9654	49.50 MB	

Tabelle 2: Ergebnis der Klassifikation aller vorhandenen Dateien des C64-Archivs.

tativen Analyse extrahiert werden können. Das Archiv enthält 502 Diskmag-Abbilder, welche 27 Diskmag-Gruppen (Verlage, welche die Disketten kommerziell vertrieben, aber auch private Organisationen) zugeordnet wurden. Eine Extraktion der Abbilder ergab, dass insgesamt 49.5 MB auf 9656 Dateien in den Abbildern gespeichert sind. Tabelle 2 gibt an, welcher Anteil an Daten/Dateien der Klassifikator den drei Klassen *unbekannt*, *komprimiert* bzw. *Text-Datei* zuordnet. Abbildung 2 zeigt, wie viele Daten die einzelnen Diskmag-Gruppen diesen drei Klassen beitragen. Diese Beobachtungen müssen als tendenziell negatives Urteil gelesen werden: aus dem C64-Archiv lässt sich wahrscheinlich kein großes Korpus an Dokumenten der 8-Bit-Szene gewinnen.

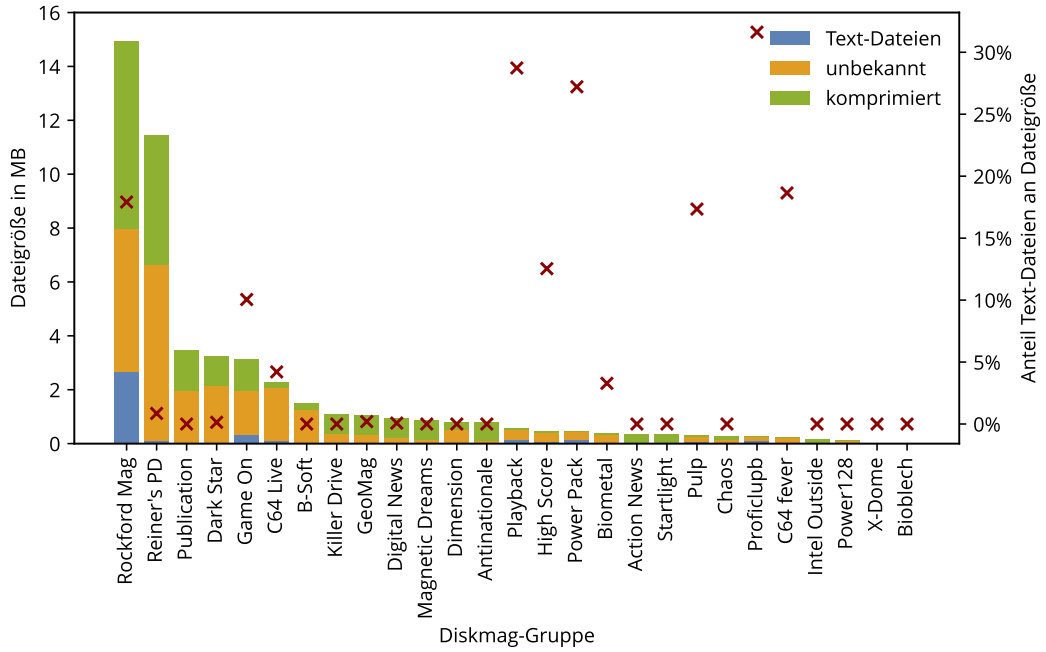

Abbildung 2: Klassifikation der Dateien im C64-Archiv gruppiert nach Diskmag-Gruppe. Der relative Anteil an Text-Dateien an der gesamten Dateigröße einer Diskmag-Gruppe ist durch die Kreuze und durch die rechte Achse gekennzeichnet.

Zum einen liegen in den Diskmags an sich relativ wenige auslesbare Text-Dateien vor. Das Programm konnte 963 Dateien bzw. 3.73 MB identifizieren. Die endgültige Extraktion liefert also zunächst ein Korpus von nur etwas 450 000 Wörtern, zum überwiegenden Teil von der von Hobbyisten herausgegebene Serie *Rockford Mag* und der kommerziellen Zeitschrift *Game On*. Gleichzeitig muss hier auch die relativ niedrige Trefferquote des Klassifikators berücksichtigt werden. Es wird durchaus fehlerhafte Klassifikationen von Text-Dateien geben, die fälschlicherweise als *unbekannt* klassifiziert wurden. Auch im Hinblick auf die im vorigen Abschnitt abgeschätzte Prävalenz im

C64-Archiv ist es wahrscheinlich, dass in den Diskmags noch einmal die selbe Menge an auslesbaren Text-Daten unerkannt vorliegen könnte. Hierfür sind verfeinerte Methoden der Klassifikation notwendig.

Zum anderen macht die Analyse deutlich, dass die komprimierten Daten ein wesentlicher Anteil an den gespeicherten Informationen auf den Diskmags haben. Unter Annahme, dass die Hälfte der komprimierten Dateien tatsächlich Text-Dateien seien, wäre ein weiterer Text-Anteil von insgesamt 28 MB in den Diskmags vorhanden.[21] Hier ergibt sich ein prinzipielles Problem: ohne genaues Wissen über das spezielle Kompressionsverfahren ist es nicht möglich, die Dateien wieder zu dekomprimieren. Zur Entwicklungszeit der Diskmags waren zwar schon einige Kompressionsprogramme in Verwendung,[22] eine klare Spezifikation der Formate (insbesondere auch die Frage ob angezeigt wird, von welchem Programm diese komprimierte Datei generiert wurde) wurde aber mutmaßlich entweder nicht angefertigt oder ist verloren. Die Dekompression müsste daher vermutlich durch aufwendiges *reverse engineering* und Analyse der Diskmag-Leseprogramme erarbeitet werden. Wie vielfältig die verwendeten Kompressionsverfahren im C64-Archiv sind, oder ob die meisten großen Diskmag-Gruppen das gleiche Kompressionsverfahren bzw. -software verwendet haben, bleibt zu untersuchen.

5. Fazit

Mit dieser Arbeit wurde ein Versuch unternommen, ein Python-Programm zu entwickeln, welche die automatische Diskmag-Textextraktion aus Disketten-Abbildern ermöglichen soll. Sowohl zu einer Vereinfachung des Editionsprozesses, als auch zum Aufbau eines größeren Textarchivs zur 8-Bit-Szene. Trotz des sehr einfachen Verfahrens ließ sich eine vergleichsweise gute Genauigkeit und Trefferquote abschätzen. Zum einen konnte durch eine Anwendung des Programms auf das Diskmag-Archiv *c64.at* Einsichten über dieses Archiv gewonnen werden. Insbesondere fällt auf, wie hoch der Anteil an komprimierten Dateien ist, und gleichzeitig wie wenig Text-Daten tatsächlich in Diskmags vorhanden sind.

Ob das Programm Editorinnen und Editoren helfen kann, effizienter Editionen von Diskmags herzustellen, bleibt zu überprüfen. Ebenso wurden auch die vom Programm aus dem Archiv gewonnenen Textdokumente nicht näher analysiert. Als Ausblick bietet es sich hier an, die Dokumente zu sichten und auf deren Strukturiertheit, Relevanz und Potenzial als Quelle für kulturgeschichtliche Untersuchungen zu bewerten.

Ferner ist das Programm mit seiner sehr einfachen Heuristik explizit nur als eine Baseline der automatischen Diskmag-Textextraktion zu verstehen. Durch Weiterentwicklungen, z.B. bei den verwendeten Verfahren, aber auch z.B. durch Aufbau und Ausnutzen eines Trainings-Datensatzes, kann die Qualität der automatischen Diskmag-Textextraktion verbessert werden. Ein weiterer offener Punkt bleibt, wie bzw. ob die dominante Portion von komprimierten Dateien auf Diskmags erschlossen werden können.

Literaturverzeichnis

C64-Wiki, s.v. »Diskettenmagazin«. Zuletzt geändert am 3. Januar 2022, besucht am 26. März 2022. https://www.c64-wiki.de/index.php?title=Diskettenmagazin&oldid=247552.

C64-Wiki, s.v. »Packer«. Zuletzt geändert am 26. Dezember 2017, besucht am 26. März 2022. https://www.c64-wiki.de/index.php?title=Packer&oldid=179893.

21. Unter der zusätzlichen Annahme, dass die Kompression von Text-Dateien die Dateigröße auf die Hälfte senken kann. Das wäre konsistent mit der Beobachtung, dass Text-Dateien eine Entropie um 4 bit pro Byte aufweisen.

22. C64-Wiki, s.v. »Packer«.

Commodore Business Machines Electronics Ltd. *Commodore 1541 Disk Drive User's Guide*. 1982. Besucht am 24. Februar 2022. https://archive.org/details/Commodore_1541_Disk_Drive_Users_Guide_1982-09_Commodore.

Gansberger, Ferdinand. »www.c64.at. Das Verzeichnis von deutschsprachigen Magazinen für den C64«. Besucht am 20. Februar 2022. https://c64.at/.

Jannidis, Fotis, Hubertus Kohle und Malte Rehbein, Hrsg. *Digital Humanities: eine Einführung*. Stuttgart: J.B. Metzler Verlag, 2017. ISBN: 978-3-476-02622-4. https://doi.org/10.1007/978-3-476-05446-3.

Jauhiainen, Tommi, Marco Lui, Marcos Zampieri, Timothy Baldwin und Krister Lindén. »Automatic Language Identification in Texts: A Survey«. *Journal of Artificial Intelligence Research* 65 (25. August 2019). https://doi.org/10.1613/jair.1.11675.

Jong, Irmen de. *cbmcodecs2*. Version 1.2. 21. Oktober 2021. Python-Paket im *Python Package Index*. Besucht am 24. Februar 2022. https://pypi.org/project/cbmcodecs2/.

Roeder, Torsten. »Magic Disk 64 – November 1987«. Zuletzt geändert am 26. Februrar 2020. Besucht am 19. März 2022. http://diskmags.elitepiraten.de/md_87-11.html.

Roeder, Torsten, und Klaus Rettinghaus. »Game On! Digitale Archäologie und Edition zu(m) Spielen«. In *Abstract zur Konferenz Digital Humanities im deutschsprachigen Raum 2020*, 138–141. DHd 2020. Spielräume: Digital Humanities zwischen Modellierung und Interpretation. 7. Tagung des Verbands »Digital Humanities im deutschsprachigen Raum«, 5. März 2020. Paderborn, 22. Februar 2020. https://doi.org/10.5281/zenodo.3666690.

Rowe, Simon J. *d64*. Version 1.6. 28. August 2021. Python-Paket im *Python Package Index*. Besucht am 24. Februar 2022. https://pypi.org/project/d64/.

VICE Team. *VICE: The versatile Commodore Emulator*. Version 3.6.1. 2022. Besucht am 24. Februar 2022. https://vice-emu.sourceforge.io/.

Walleij, Linus. »PETSCII to Unicode mapping«. Besucht am 20. Februar 2022. http://www.df.lth.se/~triad/krad/recode/petscii.html.

Wikipedia: Die freie Enzyklopädie, s.v. »Diskmag«. Zuletzt geändert am 28. November 2021. Besucht am 26. März 2022. https://de.wikipedia.org/w/index.php?title=Diskmag&oldid=217678518.

A. Programm-Optionen

```
Nutzung: read_diskmags.py [-h] [--fix-umlaute | --no-fix-umlaute]
                          [--fix-umbrueche | --no-fix-umbrueche]
                          [--xml | --html | --stdout | --writefiles [VERZEICHNIS]]
                          [-v]
                          DATEI [DATEI ...]

Lese D64-Diskmags und extrahiere Plaintext-Daten.

Argumente:
  DATEI                 Eine oder mehrere D64-Diskmag-Abbilder

Optionen:
  -h, --help            Zeige diese Hilfenachricht
  --fix-umlaute, --no-fix-umlaute
                        Ersetze für jede Datei unbelegte Codepoints mit
                        geeigneten Umlauten mittels einer automatisch
                        generierten Zuordnung (Standard: True)
  --fix-umbrueche, --no-fix-umbrueche
                        Füge für jede Datei regelmäßige Zeilenumbrüche ein,
                        falls diese fehlen (Standard: True)
  --xml                 Schreibe Ergebnisse als einzelne pseudo-TEI-codierte
                        XML-Datei in die Standardausgabe
  --html                Schreibe Ergebnisse als einzelne HTML-Datei in die
                        Standardausgabe
  --stdout              Schreibe Ergebnisse unformatiert und konkateniert in
                        die Standardausgabe
  --writefiles [VERZEICHNIS]
                        Lege für jedes Ergebnis eine Datei in VERZEICHNIS an
  -v, --verbose         Gebe ausführliche Informationen aus
```