ForTrax und Assoziativmaschine im Anwendungsfall IT-Forensik.

Besonderheiten und Leistungen dieser A-Systeme

Hans-Joachim Bentz, Andreas Dierks, Uwe Dobbratz, Matthias Glockemann, Mari Miyamoto

Imbyte gmbH, Hildesheim & imbit.net GmbH, Hildesheim

Abstract

Assoziative Technologien sind einerseits ausführlich theoretisch untersucht wie andererseits auch in einer Vielzahl von Aufgaben praktisch angewandt worden. Sie haben aber bislang in der digitalen Forensik kaum Aufmersamkeit errungen. Daher sind hier kurz die Grundlagen zusammengestellt, eine Sammlung von Eigenschaften vorgenommen und mögliche Fragen der Forensik formuliert, um die konkreten Werkzeuge *ForTrax* (ein Softwaresystem basierend auf künstlichen neuronalen Netzen) und *AM* (Assoziativmaschine, eine alternative Rechnerarchitektur mit Assoziativmatrizen) besser kennenzulernen und ihre Eignung für Forensik in der IT diskutieren zu können.

Inhalt

1. Das System ForTrax und seine Basiskomponenten

Das Trax-System besitzt diverse Komponenten, die für unterschiedliche Anwendungsfälle und unterschiedliche Datensorten eingerichtet und getestet sind. Dabei gibt es statische Teile (die eine zwar variabel wählbar aber dann fest eingestellte Matrixgröße(n) umfassen) und dynamische Versionen, die auf Datenbankanwendungen zugeschnitten sind und mit zunehmender Lernmenge mitwachsen. Beide Sorten ändern sich bei den Lernprozessen, bleiben aber in ihren Kernfähigkeiten ähnlich: Der Zugriff ist jeweils sehr schnell, die Behandlung von Eingabe-Ausgabe ist robust / fehlertolerant, mithin geeignet zur Mustererkennung. Aus technischer Sicht funktionieren sie wie sogenannte Mapping-Memories und nicht wie Listenspeicher. Die wesentlichen neuen Begriffe sind

- "Binäre Repräsentanten",
- "Spärliche Codierung",
- "Lern- und Operationsphase",
- "Fehlertoleranz" und "Robustheit",
- "Erklärbarkeit" der (jeweiligen) Ergebnisse, kurz "XAI"
- "Weiterentwicklung" durch automatische Anpassung der Codierungsfunktion(en), sowie Kontrollinstanzen der gleichen Art,
- in Assoziativschichten gespeicherte Programme und Zustandsvariable sowie
- "reduzierte Darstellung aktiver Synapsen" (RASR: reduced active synapses representation).
- Auch 3 dimensionale Systeme mit ihren "Abfrage-"/"Antwortflächen" können relevant werden.

Die Basisstruktur wird von sog. Assoziativmatrizen[1] gebildet.

> **Vorsicht**, die im Internet verwendeten Begriffe zum Themenfeld "Assoziativ" sind teilweise sehr irreführend. Das Wort "associative" wird auch im englisch-amerikanischen Sprachgebrauch für alle möglichen Strukturen und Fälle benutzt. Zum Beispiel führt https://en.wikipedia.org/wiki/Associative_memory zum (völlig andersartigen) CAM, nämlich zu https://en.wikipedia.org/wiki/Content-addressable_memory
> und auch zu https://en.wikipedia.org/wiki/Hopfield_network,
> was ebenfalls nichts nützt, wenn man die Eigenschaften der "Assoziativmatrix" verstehen, studieren oder nutzen will. Die Gewichte im Netz hier sind und bleiben binär, die Entscheidungen hingegen werden durch das **Zusammenwirken der Gewichte** gebildet, also durch die Auswertung eines ganzzahligen Schwellenwertvektors (ggf. mehrerer).

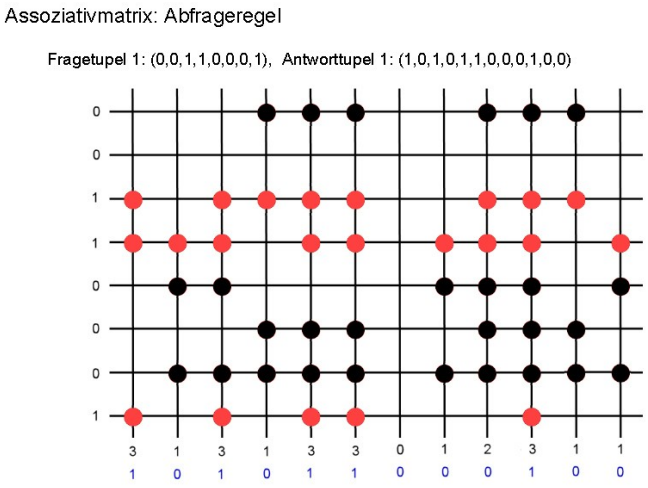


Abb. 1: Assoziativmatrix (aus: Wikipedia, siehe [1])

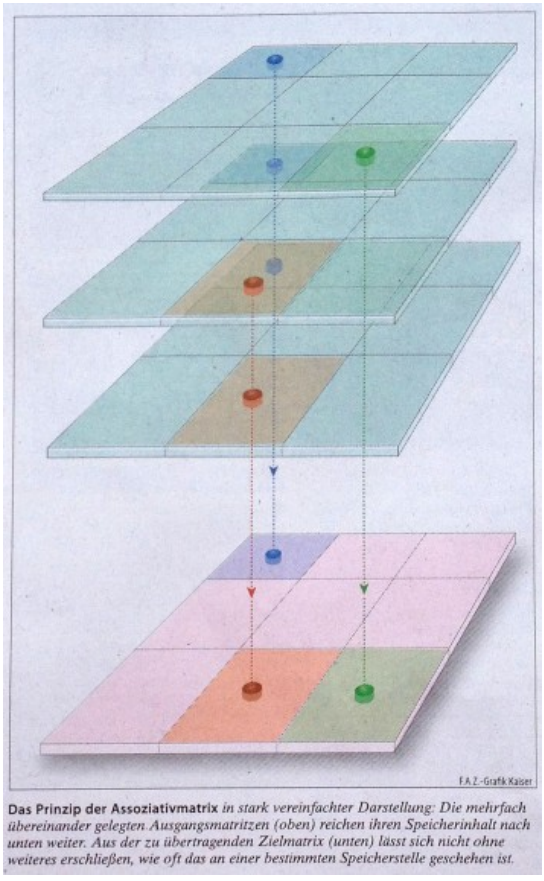
1 https://de.wikipedia.org/wiki/Assoziativmatrix

Abbildung 1 zeigt die typische Abfrageregel. Die aktiven Signale in der Anfrage, das sind die 1-Positionen im Fragevektor, führen durch die "Synapsen" (die rot markierten Verbindungsstellen) in den jeweiligen Spaltenleitungen zu einem Summierer, der die Aktivitätsstärke misst und in einem Zwischenvektor ausweist (unten notiert). Weil 3 Leitungen aktiviert wurden liegt es nahe, diesen Schwellenwertvektor mit dem Schwellenwert 3 zu reduzieren und so das angegebene Resultat als Antwort oder für die Weiterverwendung zu nutzen.

Wenn der Anfragevektor bei einer viel größeren Matrix (als in Abb. 1) lediglich ca. 1% aktive Signale enthielte, dann würde ca. 99% der Matrix nicht durchsucht werden müssen, die Abfrage mithin sehr schnell erfolgen können. Insofern spielt die **Spärlichkeit der Codierung** eines Datums eine bedeutende Rolle. In der Praxis haben sich 1-Dichten zw. 1%-5% bewährt.

Wenn man z.B. einmal (fiktiv) annähme, dass bei dem gezeigten Beispiel die Eins in Zeile 4 im Anfragevektor fehlerhaft ist, etwa auf einem Tipp- oder Übertragungsfehler beruht, dann würde man zwar nur auf der Aktivitätsschwelle 2 reduzieren, dabei aber das gleiche wie das gezeigte Ergebnis erhalten: 101 011 000 100. Insofern ist bei diesem Mechanismus typisch, dass auch Fehler ausgeglichen werden können. Bei geeigneter Wahl der wirkenden Parameter arbeitet die Assoziativmatrix bzw. ein Verbund derselben fehlertolerant und kann daher für viele Aufgaben der Mustererkennung eingesetzt werden.

Wäre das Arrangement ein Geflecht aus Hardwarebausteinen, dann würden kleinere, zufällige Fehler darin (z.B. "gekippte Bits") u.U. die Arbeitsweise nicht negativ beeinflussen. Trotz fehlender oder überzähliger Synapsen würde eine Anfrage korrekt beantwortet werden können. Diese Eigenschaft verleiht der Hardware eine Robustheit gegenüber zufälligen, lokalen Defekten.



Das Prinzip der Assoziativmatrix *in stark vereinfachter Darstellung: Die mehrfach übereinander gelegten Ausgangsmatritzen (oben) reichen ihren Speicherinhalt nach unten weiter. Aus der zu übertragenden Zielmatrix (unten) lässt sich nicht ohne weiteres erschließen, wie oft das an einer bestimmten Speicherstelle geschehen ist.*

Abb. 2: Prinzip der Assoziativmatrix[2]

Die nebenstehende Abbildung zeigt schematisch, wie aus den einzelnen Frage-Antwort Schichten die sog. Zielmatrix entsteht. Die Einsen werden dorthin projiziert und verODERt, Nullplätze bleiben Null. Das bringt es mit sich, dass man einer 1 nicht mehr ansehen kann, von welcher ursprünglichen Schicht sie stammt bzw. wie oft diese 1 in der Zielmatrix überlagert wurde. Die resultierende Struktur ist folglich sehr undurchsichtig.

Weiter kann man erkennen, vgl. auch Abb. 1, dass die gespeicherte Information (in Gestalt der Frage-Antwort Paare) nicht nur an einer Stelle im Netz abgelegt, sondern über das ganze Netz "verschmiert" ist. Das ist einer der Unterschiede zum CAM (content addressable memory).

Abbildung 1 und 2 zeigen die statischen Modelle. Bei diesen ist das Netz von vorneherein fest dimensioniert und wird in der Lernphase nach und nach befüllt. Das Netz samt Eigenschaften ist sehr gut erforscht[3], es dient einerseits als **Modell** für (natürliche) Gehirnfunktionen[4] und im technischen Bereich als **Speichermedium**[5] (s. dort speziell Ziffer 4) und **Mustererkenner**. Eine Übersicht und Einordnung in die Familie der einfachen neuronalen Netze ist ebenda zu finden.[5]

2 FAZ, Seite T4 von Dienstag 06. 05. 2014, Nr 104

3 Günther Palm: "On Associative Memory", in: "Biological Cybernetics" Nr. 36, S. 19–31, Springer-Verlag 1980

4 Günther Palm: "Neural Assemblies – An alternative approach to artificial intelligence", Springer-Vlg. 1982

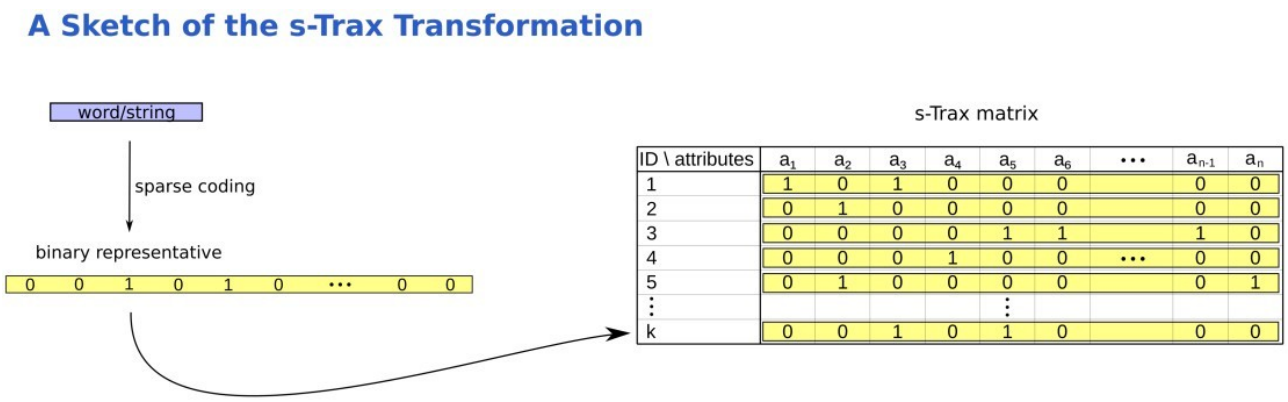


Abb. 3: Dynamische Version des A-Matrix Systems ("spärliche Codierung")

Bei den meisten Anwendungen ist allerdings das Finden geeigneter binärer Repräsentanten (Codierungen) für die realen Daten notorisch schwierig. Es zeigte sich, dass eine (generische) Modifikation der Ausgangscodierung, nämlich nur eine 1 pro Antwortvektor zu verwenden, wie in Abbildung 3, ein anderes Maximum des Informationsgehaltes hervorbringt und geeignet für den Einsatz für praktische Zwecke, meist Datenbankzugriffe, ist. ([Bentz et al. 1989][6]). Da die Matrix nun ebenfalls spärlich mit 1en besetzt ist, wählt man auch eine Modifikation der Darstellung und folglich Implementierung auf herkömmlichen Computern. Untersuchungen und Messungen zu dieser reduzierten Synapsen Darstellung (RASR, in mehreren Varianten) wurde von M. Hagström vorgenommen[7] und als praktikabel gezeigt. Die oben stehende Abbildung illustriert die Situation schematisch: Aus einem zu speichernden Wort / String werden fest-gesetzte Merkmale (patterns) p_1 bis p_m extrahiert und dazu die A-Matrix (in der Abb. "s-Trax") aufgebaut. Bei einer Anfrage aktivieren die 1-Positionen die (wenigen) zugehörigen Spalten, welche dann zeilenweise die jeweils getroffenen 1en aufsummieren und so den Zwischen-vektor, hier dann als separate Spalte, ermitteln. Bei dieser Form der Verwendung und Darstellung sind die Matrizen dynamisch, sie wachsen mit den gelernten Daten mit.

In dieser dynamischen "Befüllung" wurde der Fall analysiert, wonach z.B. frühere Items in der aktuell anstehenden Codierung berücksichtigt werden, so dass etwa Wiederholungen im Text (wie etwa ein Refrain im Gedicht) nicht zu Verwirrungen führen. Auf diese Weise kann das System sich selbst weiterentwickeln, technisch gesehen, und erhält somit neue Eigenschaften. Genaueres wurde von F. Rosenschein untersucht.[8]

Aufgrund dieser Kenntnisse und Erkenntnisse über A-Matrizen und den assoziativen Ansatz, steht für die möglichen, nachfolgend skizzierten Anwendungs- und Forschungsaufgaben der digitalen Forensik viel Know How mitsamt Basismodulen für spezielle Tools bereit.

2. Extraktion, Analyse und Sicherung von Text-Bausteinen / -Zusammenhängen

Das Trax-System besitzt mehrere unterschiedliche Such- und Analyse Funktionen. Dazu gehören speziell die *Lexico*-Suche, die *Context*-Suche, die *Treffer*-Mitteilung und die *Similar*-Suche. Zu Sicherungs- und Speicherzwecken gibt es noch den *Mindshift*-Modus, der einen (ggf. editierbaren) Schnappschuss der letzten Aktion festhält.

---

5 G. Palm & H.-J. Bentz: Theor. Unters. zu einfachen neuronalen Netzen, https://www.imbit.net/schriften/

6 HJ Bentz, M Hagström, G Palm. Information Storage and Effective Data Retrieval in Sparse Matrices. Neural Networks, 2:289–293, 1989

7 Michael Hagström. Textrecherche in großen Datenmengen auf der Basis spärlich codierter Assoziativmatrizen. (Dissertation) Universität Hildesheim 1996.

8 Fabian Rosenschein: Assoziative Systeme als Datenspeicher. (Dissertation) Universität Hildesheim 2013.

IMBYTE GmbH und imbit.net GmbH: Assoziative Technologie in der IT-Forensik. Stand 10/2023

Hier wird nur kurz auf eine Auswahl der Möglichkeiten eingegangen. Die Funktion *Lexico* zeigt zur Eingabe schreibweisenähnliche Wörter oder Begriffe im Datenbestand. So sieht man in Abb. 4 wortähnliche Begriffe zur Eingabe "Minister".

Abb. 4: Lexico-Suche zu *Minister*

Die besondere Stärke dieser Suche zeigt sich, wenn unübliche oder absichtlich verfälschte Wörter / Strings in den Daten sind, wie in Abb. 5, wo *Mohnschnecken* in den Daten auch zu *MOhnschn3ck3n* verfremdet wurde.

Abb. 5: Lexico-Suche zu *Mohnschnecken*

Das Trax-System gestattet sogar die Eingabe ganzer Sätze (inklusive eventueller Schreibfehler oder Schreibvarianten), und markiert die Teiltreffer in den Texten zur bequemeren Analyse durch den menschlichen Bearbeiter. Siehe Abb. 6.

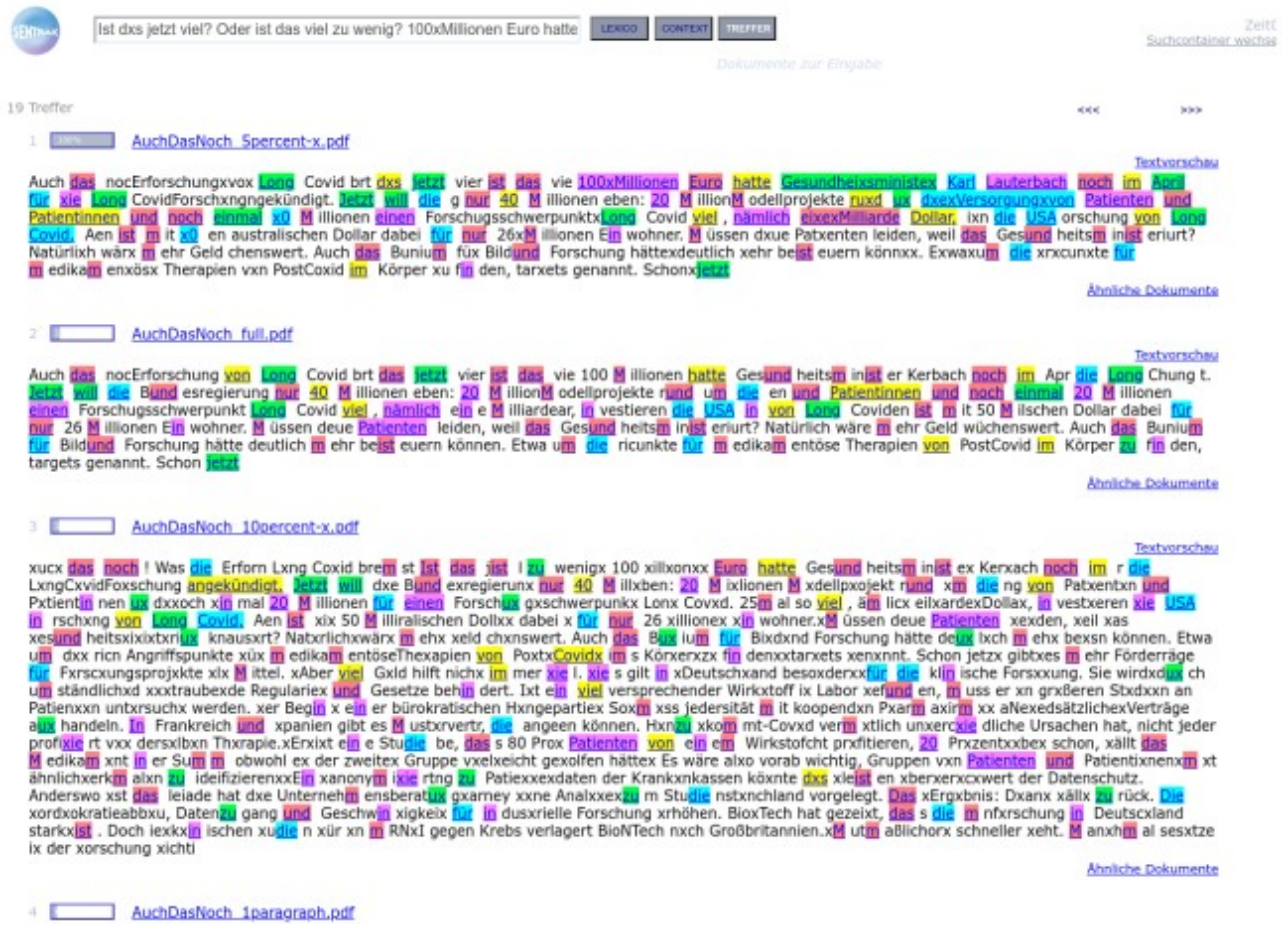


Abb. 6: Die Eingabe kann auch aus mehreren Wörtern oder ganzen Sätzen bestehen

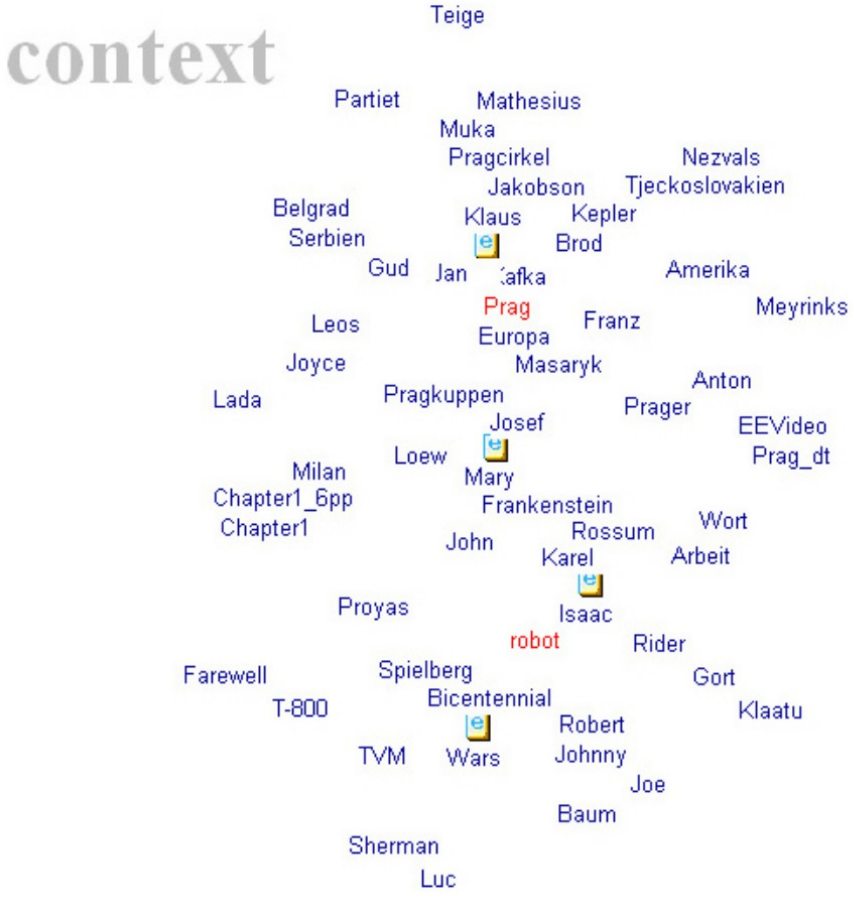


Abb. 7: Die context Map zur Eingabe *robot* und *Prag*

Die *Context*-Funktion zeigt die engsten, statistisch gemeinsamen Begriffe in den Datenblättern, bezogen auf die Eingabe.
(Die Begriffe gruppieren sich um die Dokumente, mit denen sie in besonders starker Weise in Beziehung stehen.)

Das hier ausgewählte Beispiel beruht auf einer Anfrage an eine Internetsuchmaschine, dem nachfolgenden Lernen jener Treffer (ca. 200) durch das Trax-System und der nachfolgenden Trax-Anfrage *robot Prag*.

Das Beispiel ist dem Buch von [Bentz, Dierks][9] entnommen, wo sich noch weitere derartige "Wortwolken" zu anderen Eingaben finden. Die Dichte bzw. Vielzahl der Ausgabebegriffe ist im Client-Fenster einstellbar und kann wegen der Schnelligkeit der Suche auch instantan während der Arbeit mit dem Trax-System erweitert oder reduziert werden. Die gezeigte Wortwolke besteht aus den bestpassenden Begriffen aus allen gelernten Dokumenten, was den Kontext angeht. Dabei sind in Trax immer auch die Kookkurrenzen zweiter Stufe berücksichtigt.

Mit Hilfe dieser Funktionen können Detektier- und Analysearbeiten von Aufgaben der digitalen Forensik auf automatische Weise erst ermöglicht und dann unterstützt werden. So helfen z. B. die fehlertoleranten Finde-Funktionen (z.B. *lexico*) mit, nur fragmentarisch vorliegende Text-Snippets oder willentlich verfälschtes Textmaterial überhaupt aufzufinden und zu ordnen / zu klassifizieren. Andere Funktionen, wie *context*, unterstützen das Zusammenbringen inhaltlich ähnlicher Passagen oder bedeutungsgleicher aber textuell verschiedener Notizen. Das schließt fremdsprachliche Absätze oder Einsprengsel mit ein.

Das Werkzeug ForTrax kann auf diese Weise auch absichern, dass die sog. **chain of custody** für aufgefundenes relevantes Textmaterial einen erklärbaren Anfang gewinnt, Zusammenhänge nachweist und in einer abgesicherten Umgebung bleibt. Durch die hohe Performanz der Module im Zusammenwirken mit ihren Klassifikationsfähigkeiten kann das System einen wichtigen Beitrag zur sog. **forensic readyness** (von IT-Werkzeugen) leisten.

Die vorgestellten Möglichkeiten sind vor allem für statische Gegebenheiten gedacht (auch wenn die Erkenntnisse über Textvorhandensein oder Textzusammenhänge erst nach und nach im Laufe der Arbeiten erkannt werden), weshalb dem Finden und Analysieren von Abfolgen oder Aktivitätsmustern eine eigene Untersuchung, siehe Absatz **3.**, gewidmet ist.

3. Digitale Spurenklassen, ihre Veränderung und Aktivitätsmuster

Dieser Abschnitt verweist (in diesem Stadium des Papiers) auf die redaktionelle Ausarbeitung von Dr. Andreas Dierks:
Digitale Forensik mit der AM
(3 Seiten mit Stand: 12. Februar 2023).

Assoziativmaschinen, Assoziativmatrizen, Assoziative Programmierung IMBYTE

Digitale Forensik mit der AM

Zur Analyse digitaler Spuren auf einem Datenträger sind wegen des Umfangs der zu bewältigenden Daten *automatisierte Prozesse* wünschenswert. An die Stelle KI gestützter Methoden, die in einer Lernphase viele Trainingsdurchgänge erfordern, können weit weniger aufwendige Verfahren zur Mustererkennung, -vervollständigung und -extraktion treten, die sich die Erfahrungen im Umgang mit *Assoziativmatrizen* zunutze machen, wie sie sich bei der Entwicklung von *Assoziativmaschinen* zeigen. Auf die jeweilige Größe des zu untersuchenden Datenträgers lässt sich das assoziative Verfahren schnell und einfach umstellen (Skalierbarkeit). An neue Spurenklassen und deren Änderungen passt sich das Verfahren wie vom Anwender gewünscht an (Flexibilität). Die vom assoziativen Verfahren gelieferten Ergebnisse lassen nachvollziehbar auf ihre Ursachen schließen (Erklärbarkeit).

Unser Vorgehen folgt zunächst der bekannten Abfolge bei der Untersuchung eines Datenträgers auf relevante Spuren.

In einem ersten Schritt wird das Abbild einer Festplatte oder eines anderen Datenträgers durch das Werkzeug *Plaso* (bzw. log2timeline.py) wie bei IT-forensischen Untersuchungen üblich ausgewertet (s. [Plaso GitHub], vgl. [Dewald et al.]).

Datenträger

Plaso („super timeline all the things“)

Die Plaso Werkzeuge liefern als Ergebnis ihrer Analyse eine CSV Datei (Datenbankdatei in festem Format), in der alle auf dem Datenträger vorgefundenen Ereignisse, falls ihnen eine Zeitmarkierung zugeordnet werden konnte, in ihre zeitliche Reihenfolge („super timeline“) gebracht worden sind. Um den Datenumfang nicht zu groß wer-

Veranschaulicht man sich den Inhalt einer solchen CSV Datei mit dem *Timeline Explorer* (s. [Eric Zimmerman]) so fällt auf, dass er Ereignissen Farben zuordnet, um sie hervorzuheben.

Vom *Timeline Explorer* werden unterschiedliche Arten von Ereignissen farblich hervorgehoben. In dieser Abbildung sind Programmaufrufe rot markiert.

An dieser Stelle bieten sich Assoziativmatrizen zur automatisierten Untersuchung der in der CSV Datei notierten *Timeline* an, denn die Möglichkeit des Einfärbens bedeutet, dass sich Ereignisse eindeutig erkennen und zudem in ihrer zeitlichen Abfolge zu einer *Merkmalskette* anreihen lassen.

Abb. 8: Digitale Forensik mit der AM

9 H.-J. Bentz, A. Dierks: Neuromathematik und Assoziativmaschinen, Springer Verlag, Berlin, 2013. Seite 293

4. Eine Assoziativmaschine (AM) als Kontrollinstanz

Eine AM folgt nicht dem Arbeitsprinzip der "von Neumann-Maschine", vielmehr hat sie eine ganz andersartige Architektur. Sie rechnet nicht sondern assoziiert. Operationen werden nun nicht mehr durch eine zentrale CPU gesteuert, sondern sind in einer netzartigen Struktur weit verteilt; daher erscheinen auch völlig neue Eigenschaften. Das zugrunde liegende Netz ist frei programmierbar, es wurde dazu eine neue Programmiersprache entwickelt (am1prime) und einen Simulator namens *VidAs* geschaffen, der auf herkömmlichen Computern läuft[10]. Damit lässt sich das Verhalten des Netzwerks untersuchen, dokumentieren und auch demonstrieren.

Die Anwendungsfelder dieser innovativen assoziativen Technologie sind sehr vielfältig. Ein aktuelles betrifft IT-Sicherheit. Die AM-Methodik erlaubt eine Transformation von Daten, mit der Folge, dass sowohl deren Speicherplätze (Festplatten, USB-Sticks, NAS etc.) als auch die Kommunikationskanäle beim Datenaustausch (Internet, WLAN, interne Netze etc.) abgesichert sind und dabei sogar Angriffen von Quantencomputern widerstehen.

In IT-Systeme eingebundene AM könnten auch forensiche Perspektiven bilden, die sich aus ihren Funktionsprinzipien ergeben. Dazu wären im ersten Ansatz zwei Fälle ihres Einsatzes zu unterscheiden:

(i) die AM als Kontroll- und Protokollinstrument im Netz,
(ii) die AM als Ausführende des abzusichernden Prozesses.

Für weitere Gedanken und Ausführungen zu diesem Kontext wird auf die redaktionelle Ausarbeitung von Dr. Andreas Dierks*: IT-Forensik und Assoziativmaschine verwiesen* (4 Seiten mit Stand: 24.01.2023)

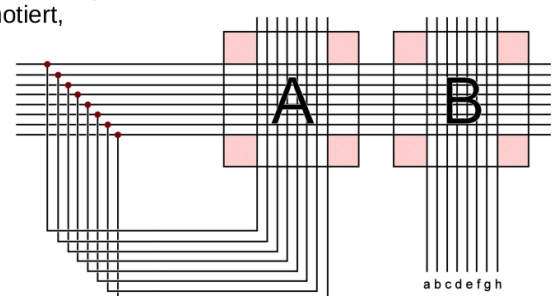
Abb. 9: IT-Forensik und Assoziativmaschine (dort Seite 4)

IMBYTE

Assoziativmaschine als Prozessor

Gedankensammlung zur Nutzung der AM in diesem Kontext:

- durch Überprüfung der Schwellwerthöhen wird erkannt und protokolliert, dass im Programmablauf Manipulationen vorgenommen wurden,
- liefert die Ablaufmatrix **A** den Wert NULL bevor der STOPPE-Befehl gegeben wurde, ist der vorgegebene Programmablauf verändert worden,
- liefert die Befehlsmatrix **B** einen Befehl, zu dem kein Befehlscode geladen wurde, wurde das Programm manipuliert,
- in einem eigenen Kurzzeitgedächtnis werden zyklisch die letzten Codes der ausgeführten Programmzeilen und Befehle protokolliert, um diese zwecks späterer Analyse abrufen zu können,
- ein und dasselbe Programm wird in mehreren nebenläufigen Prozessen, jeweils mit einem eigenen Fortsetzungskern abgearbeitet, der Vergleich der Schwellwerte von **B** weist auf missbräuchliche Nutzung hin und wird notiert,
- ...



10 Das aktuelle Simulationsprogramm VidAs 5 kann von www.assoziativmaschine.de heruntergeladen werden

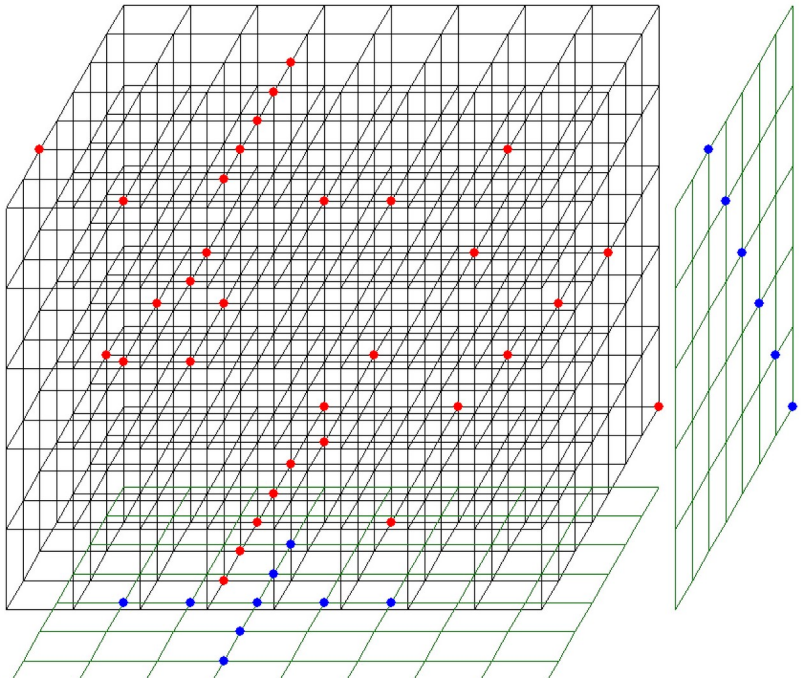


Abb. 10: Assoziative Quader

Assoziativmatrizen lassen sich geordnet in Schichten aufreihen und bilden dann z.B. einen Quader. Völlig analog zur flächigen (aber als mathematisches Objekt hochdimensionalen) Matrix lässt sich Information in diesem 3-dimensionalen Objekt speichern. Man hat es jetzt mit Frage- und Antwortflächen zu tun und kann diese separat oder gemeinsam auswerten.

Eine Veränderung der gespeicherten Information durch Angreifer ist praktisch ausgeschlossen, da jeweils die andere Antwortfläche "verzerrt" würde und damit den Eingriff signalisiert.

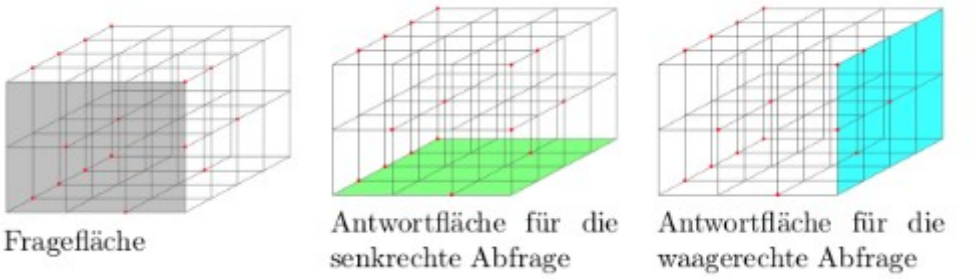


Abb. 11: Assoziative Frage- und Antwortflächen

Abb. 12: Antwortflächen mit und ohne "Störungen" (aus [Bentz, Dierks, S. 63] [9]

5. Forensik Readiness, Angriffe, Data Poisoning

Die Hinweise in Abb. 12 des vorherigen Abschnitts geben bereits eine Idee für die Überwachung von herkömmlichen komplexen NN-Systemen mittels AM. Bei dieser Aufgabenstellung sind noch weitere Ausrichtungen denkbar, ein Teil davon wird in den Folien zu IT-Forensik und Assoziativmaschine, siehe Abb. 9, angesprochen.

Umgekehrt aber kann man auch die Frage stellen, inwieweit Angriffe auf das ForTrax-System bzw. eine AM möglich sind und welche Auswirkungen diese zeigen könnten.

(i) Gewährleistung eines unveränderlichen Datensatzes.

Dieses ist grundsätzlich und auch praktisch möglich, wie etwa Abb. 10 nahelegt. Die Datensätze, die konserviert werden sollen, lassen sich z. B. in einen A-Quader einlagern und in einer der Richtungen, z.B. nach unten auch wieder auslesen. Ein einziger (~typischer) Anfragevorgang zeigt aber auch eine Antwort in die waagerechte Richtung nach rechts. Diese ist "Signatur" für die initiale / aktuelle Unversehrtheit der Daten - und lässt sich auch erklären. Jede Manipulation der Daten (diese seien auf den senkrechten, nach vorne zeigenden Flächen gespeichert) bewirkt eine Veränderung der "Signaturfläche" - und lässt sich ebenfalls erklären.

(ii) Gewährleistung einer unveränderlichen Datenmenge beim kontinuierlichen Lernen.

Die Möglichkeit, die (i) bietet, lässt sich durch regelmäßiges Sichern von Frage- und regulären Antwortflächen zusammen mit ihren "Signaturflächen" erweitern und damit quasi durch Schnappschüsse absichern. Das bedeutet, wenn zu einem Zeitpunkt im Lernprozess eine unerlaubte Veränderung oder Einschleusung erfolgt ist, dann kann dieser Zeitpunkt festgestellt und entsprechende Gegenmaßnahmen eingeleitet werden. Im schlimmsten Fall werden die Antworten als kompromittiert gewertet und ausgesondert. Aber auch dieser Fall lässt sich erklärbar begründen.

(iii) Inversionsangriffe.

Diese Art Angriff ist beim Einsatz einer AM äußerst schwierig. Die Daten aus der Realität werden repräsentiert durch große, spärlich codierte binäre Tupel (Vektoren hoher Dimension, wenn man es aus der Sicht der linearen Algebra betrachtet). Wenn Daten mit ForTrax (oder auch AM) gelernt sind, verschwinden automatisch Erkenntnisse über ein Datum, denn wer kann einem Vektor der Länge 20.000 (z.B.) mit 200 (z. B.) darauf befindlichen 1en ansehen, was er bedeutet? Insofern kann aus dem System allenfalls in umgekehrter Richtung Schlüsse gezogen werden: Ein Angreifer hat den Verdacht, dass dieses oder jenes Datum im System ist und muss nun die 1en so platzieren, dass der Verdacht bestätigt wird. Das ist aus kombinatorischen Gründen aber sehr mühsam also unpraktikabel.
Allerdings schützt das System nicht davor, dass die Daten vor dem Einlernen bereits "besichtigt" wurden.

Anmerkung: Da die Codevektoren in ein Matrixgeflecht "gelernt" werden, entweder wie im Fall der Abb. 1 und 2, oder aber in einer spärlich besetzten Matrix wie in Abb. 3, hat man als Data Owner jeweils Gegenmaßnahmen bereit. Im ersten Fall kann wegen der Überlagerung der 1en, die geODERt worden sind, nichts herausgelesen werden. Im zweiten Fall kann man die A-Matrix Spalten, die jeweils die Merkmale aufnehmen, leicht und bequem verschlüsseln. Dazu wurde eigens ein Transformationsverfahren entwickelt[11] und Prüfungen der Praktikabilität vorgenommen.

11 HJ. Bentz, A. Dierks und F. Rosenschein: **Vorrichtung und Verfahren zur Verschlüsselung;** Hildesheim 2019. Patent Nr. 10 2019 214 379 (Patentamt München)

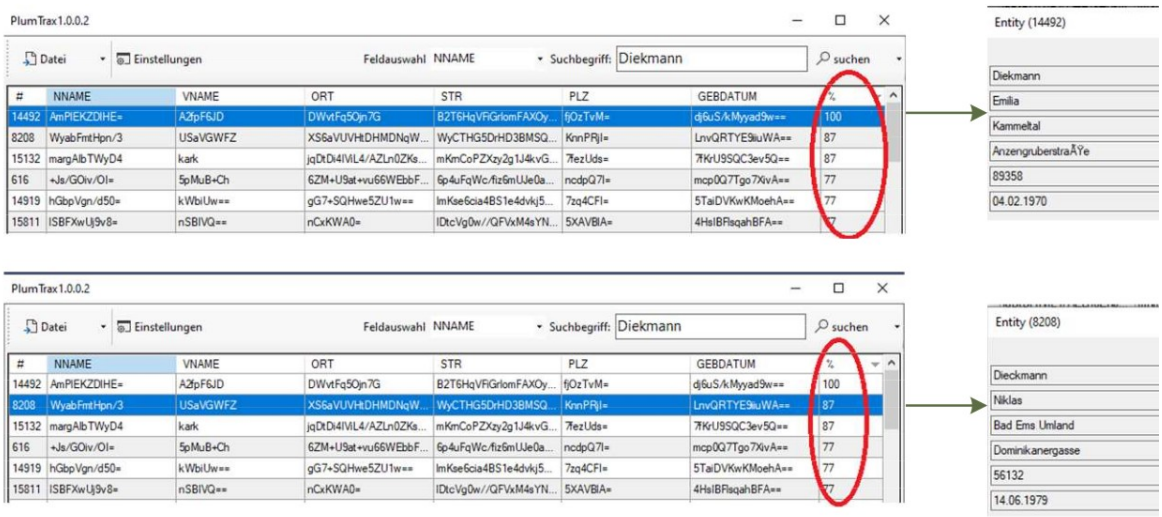


Abb. 13: Praktikabilität: Verabeitung transformierter Daten (System *plumTrax*)

Wie die Abbildung 13 andeutet, handelt es sich bei diesem Beispiel um CSV-Dateien. Diese wurden mit einer dazu passenden Codierung in ein spezielles Trax-System gelernt, wobei die Matrix spaltenweise verschlüsselt wurden (nach der aplum-Methode, siehe Fußnote 11). Dennoch kann eine **Anfrage im Klartext** gestellt werden; allerdings erscheinen die Antworten in unlesbarer Form. Um das jeweilige Kryptat zu lesen muss man den Schlüssel kennen, das System fragt danach. Im autorisierten Fall bekommt man die Feldeinträge rechts zu sehen, andernfalls bleibt es opak. Die Prüfungen wurden mit mittelgroßen Datensammlungen durchgeführt (ca 20.000 Dateien), die Operationszeiten blieben beim normalen Desktoprechner im Sekundenbereich - also praktikabel.

Insgesamt betrachtet, gibt es unterschiedliche Maßnahmen gegen diese Art von Angriffen.

Dennoch können typische **Angriffsversuche** - speziell bei Kenntnis aller internen Parameter - studiert und vorgenommen werden, um die Sicherheitsstufen sowie Abwehrmechanismen zu entwickeln und analysieren zu können.

(iv) Evasionsangriffe.

Jede (spezielle) Aufgabe erfordert eine dazu passende Codierung der ForTrax (oder AM) hinsichtlich der später zu lernenden Daten. Eventuelle Adversarial Attacker müssen das zwangsweise berücksichtigen und dementsprechend die Adversarial Inputs darauf abstimmen. Hier erscheint aber ein Widerspruch, zumindest eine Hürde, denn Adv.-Inputs zielen auf eine andere Aufgabe bzw. Ausgabe und können von der aktuellen gewählten Codierung nicht genügend stark profitieren. Es ist wie mit Mopeds und Autos: Wenn man Lasten (oder mehrere Fahrgäste transportieren will), nimmt man ein Auto. Dem Adversar hilft da kein Moped. Wenn man hingegen wendig durch enge Straßen kommen will, nimmt man ein Moped; der Adversar kann sich da nicht gut mit Autos behelfen.

(v) Data poisoning

Diese Art eines Angriffs gegen eine AM ist ziemlich schwer. Der Angreifer muss bei seiner Datenverunreinigung berücksichtigen, dass die 1-Dichte im Tupel im vorgesehenen Bereich liegt (dieser Parameter kann im System verdeckt bleiben, man muss also schon das Programm im Quellcode haben). Auch muss die relative Streuung von 1en in den Einträgen auch die Daten- (hier also Text-) Redundanzen widerspiegeln. (Diese ist bei den verschiedenen Textsorten bzw. Sprachen unterschiedlich aber typisch.) Ist das nicht der Fall, bewirkt die Gift-Vermischung keine Verschiebung der Ausgaben, da stets eine bestimmte Schwellenwerthöhe erreicht werden muss - sonst wird die Ausgabe unterdrückt oder als "Rauschen" klassifiziert.

Man erinnere sich dabei an den Umstand, dass ein Programm in der AM aus gesetzten 1en im Matrixgeflecht besteht und der Ablauf durch das Auslösen bestimmter Vektoren geschieht, die dann durch das System laufen und wiederum Aktionen (~neue Vektoren) auslösen. Diese Vektoren müssen aus Sicht des Systems bzw. Programms zulässig sein; falls nicht, stoppt es oder meldet dieses Vorkommnis. Insofern zeigt sich die Schwellenwertregelung bzw. Überwachung (die auch systemintern und parallel erfolgen kann) als großes Hindernis bei dieser Art Angriff.

Anmerkung.
F. Rosenschein hat ein System untersucht, siehe Fußnote 8 oben, das in der Zulernphase die anfangs gewählte Codierung ändert, um gewisse Wiederholungen (in den Texten) aufzufangen. Im übertragenen Sinne kann das System damit bemerken, ob es sich bei der Anfrage (oder im bestimmten Arbeitsstadium) um den ersten Refrain (z.B. eines Gedichts) oder den zweiten etc. handelt und entsprechend darauf reagieren. Im Hinblick auf Data poisoning müsste man untersuchen / erforschen, welche Art poisoning erkannt und aufgefangen werden kann oder welche Modifikationen in den Lern- und Operationsphasen vorgenommen werden könnten.

Zusammengenommen kann man sagen, dass sich ein AM-System eventuell als Kontrollinstanz für herkömmliche KNN / LLM (language learning models) einsetzen oder eben auch direkt für die forensichen IT-Aufgaben ausrichten und verwenden lässt. Es sieht so aus, dass dabei eine Reihe der Anforderungen der Forensic Readyness erfüllbar werden.

(vi) Backdoors / Hintertüren

Bei den herkömmlichen tiefen neuronalen Netzen gibt es (noch) keine Theorie dieser Systeme, wenngleich sie dem Markt sehr gelegen kommen und allseits große Faszination ausüben. Man weiß aber, dass bestimmte dieser Systeme erlauben "Backdoors" einzubauen, die nicht nachweisbar sind: *In summary, the existence of undetectable backdoors represent a significant theoretical roadblock to certifying adversarial robustness*. Mehr steht im Artikel Planting Undetectable Backdoors in Machine Learning Models[12], weiteres illustriert eine vielsagende Animation in QuantaMagazine[13]: *Cryptographers have shown how perfect security can undermine machine learning models*. Als Grund führen die Forscher gewisse Zufallseinflüsse "(for our random Fourier features backdoor)...(for our ReLU backdoor)" initial und in den Lernprozessen an, die unbemerkbares Steuern ermöglichen. Aufgrund dieser Perspektive muss man eigentlich zögern, allzuviel Aufwand zu treiben mit eigens konstruierten externen Überwachungs-Systemen für derartige Netze bzw. Methoden.

Im Unterschied zu dem impliziten Memory-Typ zB eines LLM ist bei den hier vorgestellten assoziativen Systemen das Memory völlig anders, nämlich explizit. Bei ForTrax und der AM sind die Netzstrukturen und Schichten zu Beginn völlig leer (d. h. mit Nullen besetzt). Sowohl das Laden eines Programms als auch ein Lernvorgang bewirken ein Platzieren von 1en an den relevanten Stellen, damit die in der Operationsphase durchlaufenden Vektoren/ Aktivitäten ihrerseits "Synapsen" anregen und entsprechende Zwischen-/ Ergebnisse auf den Registern als binäre Tupel ablegen. Es wirkt hier nirgendwo Zufall und ist es nicht möglich, sachfremde Programm-Module einzuschleusen; sie zerstören die Abläufe. Lediglich die spärlich codierten Repräsentanten-Paare werden zufällig bestimmt, worauf die Opaqheit des Systems beruht. Diese Art Zufall wirkt sich aber nicht auf die Arbeitsleistung oder den Funktionsumfang des Systems aus. Es ist wie bei natürlichen Gehirnen: Trotz unterschiedlicher Verknüpfung von Zellen und Nervenbahnen wissen zB. zwei Schülergehirne, dass 6x7 = 42 ist. Die Setzung der 1en einer AM ist bei jedem Neustart des Programms ebenfalls neu, die Funktionalität aber bleibt die gleiche.

Nur beim originalen Generieren eines assoziativen Programms wäre es möglich, fremde Untermodule vorzusehen und einzubauen - was aber eher auf einem Loyalitätsverstoß beruhte und nicht als fremde Attacke während der Programmabläufe gesehen werden dürfte.

12 https://arxiv.org/abs/2204.06974

13 https://www.quantamagazine.org/cryptographers-show-how-to-hide-invisible-backdoors-in-ai-20230302/

6. Zusammenfassung und Ausblick

Im Hinblick auf Forschungsfragen und Research Areas in der IT-Forensik ist Folgendes zu sagen, zuzuordnen.

A) Research Areas vom Typ Manipulationen, Angriffe, Verfälschungen: In der assoziativen Domäne sind zB. die in Abs. 5. diskutierten Angriffe nicht oder praktisch nicht möglich. Das wurde in einzelnen Punkten erläutert. Eine Stelle, "die den Zufall kontrolliert" gibt es nicht, denn es gibt keinen Zufall (wie bei den Backdoors in Fußnote 12 vorausgesetzt). Die Abläufe in einer AM können im Gegenteil auf unterschiedliche Weisen überwacht werden (z. B. durch Protokollieren der Schwellenwertdiagramme in einem sog. Langzeitgedächtnis). Dennoch können Versuche zu den diversen Angriffen durchgeführt und die Reaktionen des AM-Systems notiert, verglichen und analysiert, ggf. auch bestätigt werden.

Weil diese A-Systeme sehr schnellen Zugriff auch auf große Datenmengen erlauben (vgl. Fußnoten 6 und 7), die Fähigkeit der Klassifikation besitzen (siehe Abschnitte 2. und 3. und [D. Frobese][14] mit Güteuntersuchungen sowie [D. Frobese][15]), zudem "systemimmanent" mit einem hohen Grad an Datenschutz bzw. Privatheit ausgestattet werden können (vgl. Abb. 13), überspannen sie somit viele Anforderungen der Forensic Readyness.

Jedoch sollen die zitierten Erkenntnisse nun (erstmals) auf forensische Gegebenheiten angewendet werden. Man muss also die jeweiligen Parameter finden, erforschen, Referenzbeispiele implementieren und prüfen etc., die dann einen effizienten und gesicherten Einsatz in den Praxisfällen erlauben. Dazu gehört auch die Schaffung von Beispielen, die sich beständig weiterentwickeln. Methoden dazu wurden etwa in Fußnote 8 und auch im Umfeld zu Abb. 9 genannt. Diese Weiterentwicklung bezieht sich auch auf künftige Entscheidung von Parameterauswahlen in laufenden Lern- oder Klassifikationsprozessen durch A-Matrizen / Module. Damit ergäbe sich eine Art Hierarchie im Prozessablauf. (Als Beispiel kann man sich das Umschalten auf jeweils besser angepasste Codierungen für verschiedene Sprachen oder für Wörter mit sehr unterschiedlicher Entropie vorstellen.)

B) Forschungsthemen wie "sich weiterentwickelnde Systeme" oder "Erweiterung bestehender Methoden und Tools" oder "völlig neue Methodiken und ihre Angriffsflächen" etc.: Für diese Aufgaben sind einige Darlegungen erfolgt. Dieses schließt den Blick auf Stellschrauben wie "Aspekte für die juristischen Fragestellungen (Privatheit, Rechtssicherheit)" mit ein.

C) Da die Arbeitsschritte des assoziativen Systems, wie sie hier für "Texte" oder textuelle Daten erläutert wurden, völlig analog auch für andere Sorten Daten, wie zB. Bilder, oder sogar für Prozesse, wie zB. Steuerungsmechanismen im IoT, ablaufen, hat man mit diesem Ansatz die Möglichkeit einer Vereinheitlichung. Dazu bedarf es allerdings noch zu schaffender Soft- wie Hardwarekomponenten, die speziell auf die Felder und Aufgaben zuzuschneiden wären (sprich: geeignete spärliche Codierung vom realen Objekt/Prozess hin zum Repräsentanten im System) und die Mehrdimensionalität und Vielschichtigkeit berücksichtigen. Auf jeden Fall wäre das Verfolgen dieser Forschungs- und Entwicklungsarbeit im Sinne einer einheitlichen Abstraktion.

Stand: Oktober 2023

14 Dirk Frobese: Klassifikationsaufgaben mit der SENTRAX. Proceedings des Fünften Hildesheimer Evaluierungs- und Retrievalworkshops, Hildesheim 2006

15 Dirk T. Frobese: E-Mail-Kategorisierung und Spam-Detektion mit SENTRAX: Mustererkennung mit Assoziativmatrizen. (Dissertation). University of Hildesheim, 2009, ISBN 978-3-88120-493-4, pp. 1-176