

# Carving und semantische Analyse in der digitalen Forensik

Oren Avni, Tamara Knierim

**Abstract**—Data Carving ist das Suchen und Extrahieren von Dateien oder logischen Daten von einem Datenträger, direkt auf dem Inhalt basierend anstatt auf Metadaten. Es wird zum Beispiel benutzt, um Dateien wiederherzustellen oder versteckte Daten aufzufinden. Im Rahmen dieser Seminararbeit sollen die verschiedenen Carving-Ansätze und ihre Anwendungsbereiche untersucht und präsentiert werden. Sehr oft sind die in der digitalen Forensik zu untersuchenden Datenmengen riesig und es besteht eine Tendenz zu immer größeren Datenträgern. Weiterhin ist es schwierig, eine Methode zur Wiederherstellung von fragmentierten Dateien zu entwickeln, die präzise ist und gleichzeitig auch bei großen Datenträgern skaliert.

Zur Unterstützung der Informationsfindung bieten sich die Methoden der semantischen Analyse an. Diese Methoden werden benutzt, um Informationen eine Bedeutung zu geben und damit Lücken und Unstimmigkeiten in den vorhandenen Daten zu finden. Mit Hilfe von Ontologien wird die Untersuchung von Daten aus verschiedenen Quellen in verschiedenen Anwendungsbereichen der digitalen Forensik unterstützt, wie unter anderem der Analyse von Log-Dateien, Dokumente oder Geschäftsprozessen. Diese Arbeit untersucht Möglichkeiten und Einschränkungen der semantischen Analyse und stellt anhand mehrerer Beispiele den Bezug zwischen Theorie und Praxis dar.

**Index Terms**—Carving, semantische Analyse, digitale Forensik, Ontologien, Information Extraction, Text Mining, Reasoning, forensische Linguistik.

## I. INTRODUCTION

**C**ARVING wird in der digitalen Forensik benutzt, um Daten aus Datenträgern zu extrahieren und aus ihnen brauchbare Informationen für die Ermittlungen zu gewinnen. File oder Data Carving ist das Suchen und Extrahieren von Dateien oder logischen Daten, direkt auf dem Inhalt basierend anstatt auf Metadaten des Dateisystems, die auf den Inhalt zeigen.[25]

Carving wird in der digitalen Forensik zum Beispiel benutzt, um Dateien in nicht allokierten Gebieten [2], in Speicherausügen (Memory Dumps) oder bei beschädigten oder fehlenden Metadaten wiederherzustellen. Carving kommt weiterhin zum Einsatz wenn vermutet wird, dass die Informationen im Dateisystem manipuliert wurden.[25]

Zur Unterstützung der Informationsfindung bieten sich Methoden der semantischen Analyse an. Diese Methoden werden benutzt, um Informationen eine Bedeutung zu geben und damit Lücken und Unstimmigkeiten in den vorhandenen Daten zu finden. Mit Hilfe von Ontologien wird die Untersuchung von Daten aus verschiedenen Arten von Quellen in verschiedenen Anwendungsbereichen der digitalen Forensik unterstützt, wie u.a. der Analyse von Log-Dateien, Texten (Sprache) und Geschäftsprozessen. Im Folgenden wird der

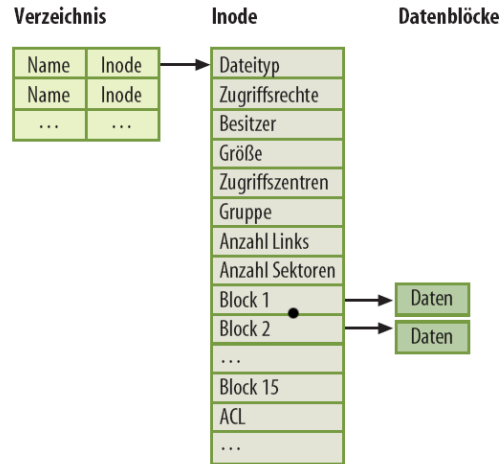


Fig. 1. Die wesentlichen Datenstrukturen von Ext 3 [23]

State of the Art der Forschungschungsarbeiten im Bereich der Anwendung der semantischen Analyse und Ontologien in der digitalen Forensik ermittelt und präsentiert.[25]

Zunächst erfolgt in Kapitel 2 eine Einführung in Carving. In Kapitel 3 werden anschließend konkrete Carving-Ansätze vorgestellt. In Kapitel 4 folgt eine Einführung in die semantische Analyse und in Kapitel 5 ein Fazit.

15. Juli 2010

## II. EINFÜHRUNG IN CARVING

Carving extrahiert Dateien von einem Datenträger, ohne dass es dabei auf die Metadaten des Dateisystems zurückgreift. Als Metadaten werden der Dateityp, die Dateigröße und die Speicherorte einer Datei und weitere Informationen, die den gespeicherten Daten zugeordnet sind, bezeichnet (siehe Abbildung 1 auf Seite 1).

Zunächst werden die Rohdaten des Datenträgers oder die Image-Datei, die ein Abbild des zu durchsuchenden Datenträgers enthält, ausgelesen. Diese Rohdaten sind die Eingabe für die Carving Methoden und werden im Folgenden als Datensatz bezeichnet. Die wiederherzustellenden Dateien müssen:

- 1) *in dem Datensatz erkannt werden*  
Carving Methoden suchen und erkennen Dateien in einem Datensatz. Dabei sind die unzähligen existierenden Dateiformate und die Fragmentierung von Dateien eine große Herausforderung.
- 2) *validiert werden*  
Nachdem im ersten Schritt eine Datei gefunden wurde, muss festgestellt werden, ob es sich tatsächlich um

eine gültige Datei handelt. Daher prüfen sog. Validierer die Datei auf bestimmte Eigenschaften. Auch Dekoder können als Validierer dienen, z.B. der JPEG Dekoder als Validierer für JPEG Bilder. Ein Dekoder kann als Validierer benutzt werden, da angenommen werden kann, dass ein Objekt gültig ist, wenn er jeden Cluster in der Sequenz richtig dekodieren kann.[2] Eine Herausforderung bei der Validierung sind die unzähligen existierenden Dateiformate.

### 3) einem menschlichen Experten präsentiert werden

Zum Abschluss werden die Ergebnisse geprüft, wobei False-Positives herausgefiltert werden. False-Positives sind Dateien, die validiert wurden, aber dennoch nicht gültig, bzw. so nicht zu verwenden sind. Diese Dateien müssen evtl. manuell nachbearbeitet werden.

Carving wird eingesetzt wenn Dateisystem-Strukturen fehlen oder beschädigt sind, wenn befürchtet wird, dass das Dateisystem manipuliert wurde, oder um Dateien an Orten zu finden, auf die ein Benutzer keinen direkten Zugriff hat, wie z.B. [14]:

- *in nicht allokierten Gebieten*
- *in Clustern, die in den Metadaten als defekt markiert wurden*
- *in der Host Protected Area (HPA)*  
Die HPA, die auch als Hidden Protected Area oder ATA-geschützter Bereich bekannt ist, ist ein reservierter Bereich für die Speicherung von Daten außerhalb eines Dateisystems. Sie wurde entwickelt, um Informationen so speichern zu können, dass sie nicht einfach durch Benutzer, das BIOS oder das Betriebssystem verändert oder von Benutzern eingesehen werden können. Die HPA kann etwa HDD Utilities, Diagnose-Tools, Bootsektor-Code und Daten für eine Systemwiederherstellung enthalten. [1]
- *im Device Configuration Overlay (DCO)*  
Das DCO ermöglicht Systemanbietern, Festplatten von verschiedenen Herstellern mit möglicherweise unterschiedlichen Größen zu kaufen und anschließend alle Festplatten auf die gleiche Anzahl von Sektoren zu konfigurieren. Ein Beispiel hierfür wäre eine 80GB HDD mit Hilfe von DCO sowohl für das Betriebssystem als auch für das BIOS wie eine 60GB HDD erscheinen zu lassen.[1]
- *im File-Slack*

Die für Datenträger kleinste beschreibbare Dateneinheit sind Sektoren. Dateisysteme wiederrum allokierten für Dateien je nach deren Größe eine gewisse Anzahl an Clustern. Ein Cluster ist die kleinste Dateneinheit, die Dateisysteme allokierten, und besteht aus einem bis mehreren Sektoren (siehe Abbildung 2 auf Seite 2). File Slack ist der Bereich zwischen dem Ende einer Datei und dem Ende des letzten Clusters dieser Datei. Wenn beispielsweise eine Datei von 1 Byte Größe auf einer NTFS Partition mit einer Clustergröße von 4096 Byte gespeichert wird, belegt diese Datei ein ganzes Cluster. Diese Datei hat dann einen File-Slack von 4095 Byte.[1]

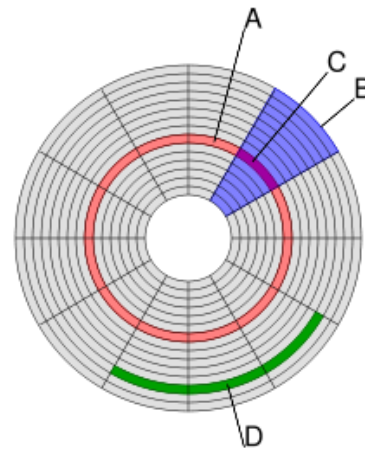


Fig. 2. Struktur einer Festplatte: (A) Track (B) geometrischer Sektor (C) (Track/physischer) Sektor (D) Cluster [30]

- *im Partition-Slack*

Partition Slack entsteht, wenn eine logische Partition nicht perfekt in die physikalischen Unterteilungen der Festplatte passt, nämlich wenn die Anzahl der Sektoren eines Datenträgers kein Vielfaches der Clustergröße sind.[22]

Auch bereits gelöschte Dateien können eventuell wiederhergestellt werden. Bei vielen Dateisystemen werden die Dateien lediglich in den Metadaten als gelöscht markiert und die Datei nicht explizit auf dem Datenträger gelöscht.[2]

Carving birgt einige wichtige Herausforderungen in sich, die im Folgenden erläutert werden. So müssen die Eigenheiten unzähliger Dateiformate beachtet und bei Fragmentierung die Fragmente der Dateien richtig zusammengesetzt werden. Dies kann aufgrund der großen Datenmengen und der hohen Anzahl an Rechenschritten sehr viel Zeit in Anspruch nehmen. Weiterhin stellt sich die Frage wie die Qualität der Ergebnisse von Carving Tools festgestellt werden kann und ob die Ergebnisse vor Gericht verwendet werden dürfen.

#### A. Dateiformate

Eine Herausforderung für das Carving sind die unzähligen Dateiformate, die auf den Datenträgern vorkommen und hier erkannt werden müssen. Wird eine Datei von einem Carving Algorithmus erkannt, muss sie validiert werden, d.h. es muss festgestellt werden, ob die Datei intakt, bzw. gültig ist. Diese Tätigkeiten erfordern spezielle, fundierte Kenntnisse über jeden Dateityp.[28]

Häufig treten sog. False-Positives auf, d.h. es werden Dateien validiert, die nicht gültig sind. Diese Dateien können beispielsweise unvollständig sein oder es können sich nicht zur Datei gehörende Daten darin befinden. Die von einem Carving Tool gefundenen Dateien müssen daher abschließend von einem menschlichen Experten begutachtet und evtl. manuell nachbearbeitet werden.

Meist sind diejenigen Dateien für die forensische Verwertung von Bedeutung, die von Benutzern erstellt und modifiziert

werden (z.B. DOC, JPEG, PST), wohingegen Systemdateien (z.B. INF, HELP oder INI) von geringerem Interesse sind.[28]

### B. Fragmentierung

Fragmentierte Datenträger sind eine besondere Herausforderung für das Carving. Fragmentierung bedeutet, dass Dateien nicht kontinuierlich auf der Platte gespeichert werden, wodurch die Zeit zum Lesen der Datei signifikant ansteigen kann. Je länger die Lesezugriffszeit des Datenträgers ist, desto stärker wirkt sich eine Fragmentierung auf die Systemperformance aus [28]. Die Fragmente von fragmentierten Dateien müssen beim Carving gefunden und richtig zugeordnet werden.

Fragmentierung kann mehrere Ursachen haben [28]:

- Wurde eine Datei vergrößert, kann es sein, dass sie zusätzliche Cluster allokiert muss. Ist der nachfolgende Cluster schon belegt, wird die Datei fragmentiert.
- Wird eine Datei gelöscht oder verkleinert, entsteht eine Lücke. Diese Lücke kann für die nächste Datei zu klein sein, woraufhin sie fragmentiert wird.
- Besonders wenn geringer Speicherplatz vorhanden ist, werden auch kleine Lücken genutzt und so entsteht eine stärkere Fragmentierung.
- Zudem können Eigenheiten einiger Dateisysteme zur Fragmentierung beitragen.

Ein Beispiel hierfür ist die Block Suballocation [18] (oder auch Block Level Fragmentation, Tail Merging or Tail Packing genannt), die von einigen Unix Dateisystemen wie z.B. UFS eingesetzt wird. Durch sie soll der Verlust von Speicherplatz minimiert werden. Hier wird der letzte Teilblock der Daten von verschiedenen Dateien in einem einzigen Cluster gespeichert, anstatt in mehreren, dann zum größten Teil leeren Clustern.[16]

S. L. Garfinkel [28] veröffentlichte 2007 eine Studie über Dateifragmentierung in der Praxis (real-world), in deren Rahmen über 300 gebrauchte Festplatten ausgewertet wurden. Die Festplatten wurden zwischen 1998-2006 gesammelt, viele wurden bei eBay erworben. 6% der Dateien auf diesen Festplatten waren fragmentiert. Die Studie zeigte weiterhin, dass sich die verschiedenen Dateisysteme und auch die unterschiedlichen Dateiformate in der Häufigkeit der Fragmentierung unterscheiden. Die für forensische Zwecke besonders interessanten Dateiformate (z.B. AVI, DOC, JPEG and PST) wiesen eine signifikant höhere Fragmentierungsrate als die weniger relevanten Dateiformate (BMP, HLP, INF and INI) auf. Eine Ausnahme bildeten hier allerdings die DLL und CAB Systemdateien, die wahrscheinlich durch Systempatches und -upgrades stark fragmentiert waren. Bekannte DLL und CAB Dateien könnten allerdings mit Hilfe einer Datenbank, die Hashcodes von DLL und CAB Dateien enthält, aus dem Datensatz herausgefiltert werden. [28]

Moderne Dateisysteme versuchen eine Fragmentierung üblicherweise zu vermeiden, da es länger dauert fragmentierte

Dateien zu lesen und zu schreiben als nicht-fragmentierte Dateien [28]. Solid state disks (SSDs) beispielsweise, sind hier aber eine Ausnahme. SSDs basieren auf Halbleiterspeicherbausteinen und haben daher sehr geringe Lesezugriffszeiten. Hinzu kommt, dass bei Flash-SSDs eine Flashzelle in ihrem Leben nur bis zu 100.000 Schreibvorgänge absolvieren kann. Daher verwenden die Hersteller sog. Wear-leveling Algorithmen, die bewirken, dass jede Zelle möglichst gleich häufig beschrieben wird [20]. Wear-leveling Algorithmen führen zu einer starken Fragmentierung. Die hierfür verwendeten Algorithmen sind herstellerepezifisch. [29]

### C. Zeitaufwand

Carving ist eine komplexe Aufgabe und erfordert auch durch die Trends in der digitalen Forensik eine immer stärkere Rechenleistung. Diese Trends sind eine Zunahme der durchschnittlichen Größe (gemessen in Bytes) der Datenträger, eine Erhöhung der Anzahl an Fällen, in denen digitale Forensik eingesetzt wird, und die Entwicklung von Tools der nächsten Generation, die mehr Rechenkapazität benötigen. Als Folge müssen Entwickler in der digitalen Forensik alle zur Verfügung stehenden Mittel nutzen, um die Leistungsfähigkeit der Instrumente zu erhöhen.

Ein Mittel ist der Einsatz von Software, die die Vorteile moderner Multicore-CPUs mit Hilfe von Multithreading nutzt. Dabei kann ebenfalls eine Nutzung von GPUs (Graphical Processor Units) erwogen werden. [21] Das Carving Tool Forensik Toolkit bietet darüber hinaus verteilte Datenverarbeitung an [12].

Um die Laufzeit zu verkürzen wird weiterhin nach effizienten Algorithmen geforscht. Weiterhin wird versucht, die zu durchsuchenden Daten sinnvoll einzugrenzen: Wenn noch ein intaktes Dateisystem vorhanden ist, und nicht davon ausgegangen wird, dass es manipuliert wurde, ist es beispielsweise eventuell ausreichend nur die nicht allokierten Gebiete der Platte zu analysieren. Nicht allokierte Gebiete sind die Gebiete des Datenträgers, die nach Angabe der Metadaten des Dateisystems keine Datei-Informationen enthalten [2]. Eine weitere Möglichkeit, die zu durchsuchenden Daten einzugrenzen, ist, bekannte Systemdateien anhand von Hashcodes gezielt aus dem Datensatz herauszufiltern [28].

### D. Qualitätsaspekte

Die Ergebnisse eines Carving Tools können einen erheblichen Einfluss darauf haben, welche Informationen für die Ermittlung einer Straftat zur Verfügung stehen. Wenn ein Tool zu guten Ergebnissen führt, können wertvolle Informationen aufgedeckt werden. Wie aber kann die Qualität von Carving Tools objektiv beurteilt werden? Wichtige Fragen bei der Beurteilung sind [27]:

- 1) *Wie viele relevante Daten werden nicht wiederhergestellt?*

Informationen, die von einem Tool nicht wiederhergestellt wurde, werden höchstwahrscheinlich ignoriert, da die Datensätze der Untersuchung viel zu groß für die manuelle Inspektion sind. Daher ist es wichtig, dass

ein Tool so viel nützliche Informationen wie möglich extrahiert.

2) *Wie präzise ist ein Tool, d.h. wie viele wiederhergestellten Daten sind korrekt?*

Die Aufdeckung von möglichst vielen Informationen kann wiederum ebenfalls zu Problemen führen. Wenn viele falsche Ergebnisse (False-Positives) produziert werden, wie unlesbare Dokumente, so muss ein Ermittler diese falschen Ergebnisse manuell überprüfen. Nicht nur, dass dies sehr viel Zeit kostet, sondern es werden auch Beweise übersehen oder die Ergebnisse eines Tools werden komplett verworfen.

3) *Wie viele Daten, die das Tool angibt wiederherstellen zu können, stellt es tatsächlich wieder her?*

Ein Tool muss zuverlässig sein. Wenn ein Tool eine Datei nicht wiederherstellt, weil es ihren Dateityp nicht unterstützt, ist das zwar schlecht, aber ein Ermittler weiß zumindest, dass Dateien dieses Typs werden nicht wiederhergestellt werden. Dateien, die offiziell unterstützt werden, aber nicht wiederhergestellt werden, können einen Ermittler zu der Annahme verleiten, dass diese Dateien nicht im Datensatz vorhanden sind.

Bei der Bewertung eines Carving Tools auf Basis seiner Ergebnisse gibt es auch Nachteile. Das größte Problem ist, dass die Ergebnisse eine Kombination von beidem, dem Tool und dem zu untersuchenden Datensatz sind. Ein Tool kann bei der Durchführung mit bestimmten Datensätzen zu sehr guten Ergebnissen führen und bei anderen Datensätzen zu sehr schlechten Ergebnissen führen. [27]

### E. Verwendbarkeit der Carving-Ergebnisse vor Gericht

Damit Ergebnisse, die mit Hilfe von Carving gewonnen wurden, vor Gericht verwendet werden können, muss zunächst auf eine korrekte Vorgehensweise während der gesamten Ermittlung geachtet werden (siehe z.B. Einführung in die Computer Forensik von Paul Baumann und N. Rahn [24] und Hostforensik von P. Neugebauer und T. Volk [26]). Beispielsweise sollte jeder Schritt dokumentiert werden. Vor Beginn eines Carving im Rahmen einer digitalen Forensik, muss eine Spurensicherung stattfinden. Ein zu durchsuchender Datenträger wird vor dem Carving entweder bitweise 1:1 gesichert oder es sollte ein Write Blocker angebracht werden, der verhindert, dass die Daten auf dem original-Datenträger verändert werden (siehe auch Hostforensik [26] Extraktion von Speicherinhalt auf S. 63-70). Vor Gericht sind weiterhin nicht nur die gesicherten Daten selbst relevant, sondern auch die Aussagen des Ermittlers, der diese Daten gesichert und ausgewertet hat. [26]

Die Rechtsprechung des Bundesverfassungsgerichts verlangt zur rechtssicheren Beweismittelsicherung und -auswertung, dass:

- Verhältnismäßigkeitsgrundsätze und Verfahrensrechte eingehalten werden [7]
- eine übermäßige Datenerhebung vermieden wird [7]

- die Erhebung kernbereichsrelevanter Daten (es gibt einen absolut geschützten Kernbereich privater Lebensgestaltung), soweit informationstechnisch und ermittlungstechnisch möglich, unterbleibt [6]

Daher kann eine Beschränkung auf eine teilweise Sicherung statt eine 1:1 Kopie erforderlich sein [26]. Bei schwerwiegenden, bewußten oder willkürlichen Verfahrensverstößen ist ein Beweisverwertungsverbot geboten. [7]

Aber auch die Hersteller der bekannten Carving Tools Encase Forensic [15], Forensic Toolkit [11] und X-Ways Forensics [31] bemühen sich darum, dass die Ergebnisse ihrer Tools vor Gericht verwendet werden können. Encase und Forensic Toolkit wurden beispielsweise nach eigenen Angaben in den USA vom dortigen Gerichtshof validiert. X-Ways Forensics ist ein Carving Tool eines deutschen Herstellers und gibt auf seiner Webseite an, durch dosierte und selektive Sicherungs-, Ausblende- und Filtermöglichkeiten und rechtssichere Beweismittelsicherung und -auswertung besonders gut für die deutsche Rechtsprechung geeignet zu sein.

### III. CARVING-ANSÄTZE

Im Folgenden wird ein Überblick über bestehende Carving Ansätze gegeben.

#### A. File Structure Based Carver

Carving Tools der ersten Generation verwendeten File Structure Based Carving Techniken. File Structure Based Carver sind Carver, die Bytesequenzen benutzen, um Dateien zu erkennen und wiederherzustellen. Sie suchen zunächst nach einem Datei-Header, einer sogenannten Magic Number. Ein Header identifiziert den Dateibeginn, er ist eine Identifikationssequenz, die für jeden Dateityp einzigartig ist. File Structure Based Carver nutzen dies um im Datensatz den Beginn von Dateien zu erkennen. So beginnt beispielsweise ein JPEG Bild mit den Byte Werten 0xFF D8 (Header) [28]. Die Konstante 0xFF D8 wird auch als magic number bezeichnet. Byte-für-Byte Vergleiche sind eine der schnellsten Operationen, die moderne Computer ausführen können [28].

Unterschieden wird hauptsächlich zwischen folgenden Techniken [10]:

- *Header/Footer Carving*  
Eine Methode, die Dateien mittels ihres Headers und Footers aus den Rohdaten extrahiert.  
So beginnt beispielsweise ein JPEG Bild mit den Byte Werten 0xFF D8 (Header) und endet mit den Byte Werten 0xFF D9 (Footer) [28].
- *Header/Embedded Length Carving*  
Eine Methode, die Dateien mittels ihres Headers und einer Länge (Größe), die im Dateiformat eingebettet ist, aus Rohdaten extrahiert.
- *Header/Maximum (file) size Carving*  
Eine Methode, die Dateien mittels ihres Headers und einer maximalen (Datei) Größe aus Rohdaten extrahiert.

Header/Maximum file size Carving sucht nach einem Header und übergibt alle restlichen Daten des Datensatzes ab dem Header an einen Validierer weiter. Es kann angewendet werden, wenn einen Validierer nur Daten zwischen einer Anfangsmarke und einer Endmarke interessieren und er angehängte, nicht zur Datei gehörende Daten ignoriert, wie das auch bei einem JPEG Dekoder der Fall ist [28].

- *Block Based Carving*

Jede Methode, die Rohdaten auf einer Block-für-Block Basis analysiert, um zu bestimmen, ob ein Block ein Teil einer möglichen Ausgabedatei ist.

Bei Block Based Carving werden jeweils nur die ersten Bytes eines Blocks mit bekannten Headersequenzen verglichen [28]. Cluster als auch Sektoren können auch als Block bezeichnet werden. Block Based Carving setzt voraus, dass jeder Sektor nur Teil einer einzigen Datei (oder eingebetteten Datei) ist.

- *Character Based Carving*

Jede Methode, die Rohdaten auf einer Character-für-Character Basis analysiert, um zu bestimmen, ob ein Sektor ein Teil der möglichen Ausgabedatei ist.

Character Based Carving findet auch Objekte, die in Container-Dateien eingebettet sind und deren Header sich nicht am Anfang eines Sektors befindet. Character Based Carving ist auch notwendig, um Dateien wiederherzustellen, die unter Dateisystemen wie ReiserFS abgespeichert wurden, die sich beim Speichern von Dateien nicht auf Sektorengrenzen beschränken [28].

- *Fast Object Validation/Carving*

S. L. Garfinkel [28] schlägt ein Vorgehen vor, das er Fast Object Validation nennt, um Dateien möglichst schnell zu extrahieren. Ein guter Carver kann die überwiegende Mehrheit der Byte-Sequenzen in einem Datensatz ausschließen, ohne zu versuchen, diese zu überprüfen. Wenn beispielsweise bekannt ist, dass Dateien, bzw. Identifikationssequenzen (Header) nur am Anfang von Sektoren starten können, müssen bei einer Sektorengröße von 512Bytes  $511/512 = 99,8\%$  der Strings nie verglichen werden (Block Based Carving).

Wird ein Dateiheader gefunden, werden (wie auch beim Header/Maximum (file) size Carving) ab dem Header alle restlichen Daten des Datensatzes an einen Validierer weitergegeben. Erst wenn ein solcher Datenteil validiert, stellt der Carver die tatsächliche Länge der Datei fest. Ebenfalls wie Header/Maximum (file) size Carving kann Fast Object Carving nur angewendet werden, wenn einen Validierer nur Daten zwischen einer Anfangsmarke und einer Endmarke interessieren und er angehängte, nicht zur Datei gehörende Daten ignoriert.

Die ersten Carving Tools benutzten Header/Footer Carving und Header/Embedded Length Carving. Foremost, das von den United States Air Force Office of Special Investigations entwickelt wurde, war eines der ersten Carving Tools, die Header/Footer Carving sowie Header/Embedded Length Car-

ving durchführten. [2]

Zur Reduzierung des wiederherzustellenden Datenvolumens bieten eine Reihe von forensischen Anwendungen die Option Dateien, die in Betriebssystemen und Anwendungen weit verbreitet sind, aufgrund ihrer MD5 Hashes und Keywords zu identifizieren. [2]

File Structure Based Carver haben allerdings zwei entscheidende Nachteile [28]:

- 1) *sie können keine fragmentierten Dateien wiederherstellen*

- File Structure Based Carver können lediglich zusammenhängende Dateien wiederherstellen. Sie extrahieren sequenzielle Bytes von dem zu durchsuchenden Datenträger. Daher enthalten die Dateien, die diese Carver als Ergebnis zurückgeben, häufig zusätzliche, nicht zur Datei gehörende Daten.

- Laut der Studie von S. L. Garfinkel [28] (siehe Kapitel II-B), sind in der Praxis etwa 6% der Dateien fragmentiert und können daher mit File Structure Based Carver nicht wiederhergestellt werden.

- 2) *sie nehmen keine ausreichende Validierung vor*

- Es wird keine umfangreiche Validierung der Dateien durchgeführt. Somit enthalten die Ergebnisse von File Structure Based Carver viele False-Positives.

### B. Fragment Recovery Carving

Fragment Recovery Carving wurde 2006 von S. L. Garfinkel [28] entwickelt, der diese Vorgehensweise auch Split Carving nannte. Fragment Recovery Carving ist eine Carving-Methode, die Dateien wiederherstellen kann, die in zwei Fragmente aufgeteilt sind.

#### Bifragment Gap Carving

Bifragment Gap Carving testet alle Kombinationen von Clustern zwischen einem Header und Footer (unter der Annahme, dass nur eine Lücke existiert), bis eine erfolgreiche Decodierung/Validierung möglich ist.

Der Algorithmus [28] (siehe Abbildung 3 auf Seite 6):

- Sei  $f_1$  das erste Fragment das von Sektor  $s_1$  bis  $e_1$  reicht und sei  $f_2$  das zweite fragment das von Sektor  $s_2$  zu  $e_2$  reicht.
- $g$  sei die Größe der Lücke (Gap) zwischen zwei Fragmenten, nämlich  $g=s_2-(e_1+1)$
- $s_1$  und  $e_2$  sind (aufgrund des vorhandenen Headers und Footers) bekannt, der Carver muss  $e_1$  und  $s_2$  finden
- Angefangen mit  $g=1$ , teste (mit einem Validierer) alle möglichen Positionen der Lücken bis  $g=e_2-s_1$
- Für jedes  $g$ , teste (mit einem Validierer) alle konsistenten Werte von  $e_1$  und  $s_2$ .

Bifragment Gap Carving führt zu zufriedenstellenden Ergebnissen, wenn die beiden Fragmente nahe beieinander liegen. Allerdings hat es die folgenden Einschränkungen [2]:

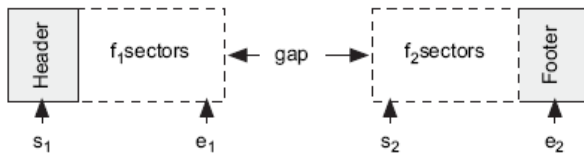


Fig. 3. Bifragment Gap Carving [28]



Fig. 4. Validiertes, aber dennoch inkorrektes JPEG Bild aus dem DFRWS 2007 Testset [2]

- Bifragment Gap Carving skaliert nicht für Dateien mit großen Lücken. Wenn  $n$  die Anzahl der Cluster zwischen  $e_1$  und  $s_2$  sind, sind im schlechtesten Fall  $n^2$  Validierungen für eine Wiederherstellung nötig.
- Bifragment Gap Carving wurde nicht für Dateien mit mehr als zwei Fragmenten ausgelegt.
- Erfolgreiches dekodieren/validieren impliziert nicht immer dass die Datei korrekt wiederhergestellt wurde (siehe Abbildung 4 auf Seite 6).
- Bifragment Gap Carving funktioniert nur mit Dateien, die eine Struktur haben, die validiert/dekodiert werden können.
- Bei fehlenden oder korrupten Clustern tritt oft der schlechteste Fall ein.

#### Bifragment Carving with constant size and known offset

Microsoft OLE (Object Linking and Embedding) Dokumente wie z.B. Word oder Excel, die in zwei Fragmente aufgeteilt sind, können mit Bifragment Gap Carving nicht wiederhergestellt werden, da sie keinen erkennbaren Footer haben. Bifragment Carving with constant size and known offset nutzt im CDH (Compound Document Header) und in der MSAT (Master Sector Allocation Table) enthaltene Informationen, um MSOLE Dateien wiederherzustellen. Die CDH befindet sich im ersten Sektor der Datei und enthält einen Zeiger, der auf die MSAT zeigt. Die MSAT befindet sich tendenziell nahe dem Dateieende und daher meist im zweiten Fragment. Auch hier gilt die Einschränkung, dass keine Dateien mit mehr als zwei Fragmenten wiederhergestellt werden können. [28]

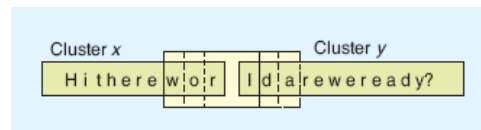


Fig. 5. PPM: Gewicht auf Basis eines Schiebefenster der Größe vier berechnen [2]

#### C. Graph Theoretic Carver

Graph Theoretic Carver [2] sind ein weiterer Ansatz fragmentierte Dateien wiederherzustellen. Jeder Cluster wird von einem Knoten des Graphen repräsentiert. Gegeben sei eine Menge von nicht allokierten Clustern  $b_0, b_1, \dots, b_n$ , die zu einem Dokument  $A$  gehören. Gesucht ist diejenige Permutation  $P$ , die die Struktur des ursprünglichen Dokuments repräsentiert. Um die korrekte Reihenfolge der Cluster zu bestimmen, müssen Fragmentpaare identifiziert werden, die im ursprünglichen Dokument adjazent (benachbart) sind. Dazu wird zunächst zwischen jedem Knotenpaar  $b_x$  und  $b_y$  eine Kante erstellt, die mit einem Gewicht versehen wird ( $W_{x,y}$ ). Das Gewicht gibt die Wahrscheinlichkeit an, dass Cluster  $b_y$  auf Cluster  $b_x$  folgt. Es gibt nun viele Möglichkeiten, die Gewichte zwischen zwei Clustern zu bestimmen. Im Folgenden werden zwei bekannte Ansätze erläutert:

- *Gewichte bestimmen mit Hilfe von PPM (prediction by partial matching):*

PPM ist eine Kontext-Modellierungstechnik mit endlichen Folgen, die von Cleary und Witten [17] eingeführt und zum Benchmark für verlustfreie Datenkompressionstechniken wurde. Das Gewicht ( $W_{x,y}$ ) zwischen Cluster  $b_x$  und Cluster  $b_y$  wird durch Verschieben eines Fensters über die Endbytes des Cluster  $b_x$  und die Anfangsbytes des Cluster  $b_y$  berechnet. Abbildung 5 auf Seite 6 zeigt, wie das Gewicht auf Basis eines Schiebefenster der Größe vier berechnet wird: Das Fenster wird zunächst an den Rand des Clusters  $b_x$  gesetzt und die Wahrscheinlichkeit berechnet, dass "l" nach "wor" kommt. Anschließend gleitet das Fenster ein Byte weiter und es wird die Wahrscheinlichkeit berechnet, dass "d" nach "orl" folgt. Dieses Ergebnis wird mit der vorherigen Wahrscheinlichkeit ("wor"- "l") multipliziert. Dieser Vorgang wird wiederholt, bis sich das Fensters nicht mehr in Cluster  $b_x$  befindet. Dies ergibt das Gewicht ( $W_{x,y}$ ). Während PPM für strukturierte Daten, wie Text, gut geeignet ist, ist es für Grafiken und komprimierte Dokumente weniger gut geeignet. [2]

- *Gewichte bestimmen im Fall von 24-b Windows BMP Grafiken:*

Pal et al. [4] entwickelten eine Methode für die Wiederherstellung von fragmentierten Bilddateien. In ihrer Forschung verwendeten sie 24-b Windows BMPs, die stark fragmentiert waren. Um das Gewicht einer Kante zwischen zwei Clustern zu bestimmen, verglichen sie die Grenzen zwischen jedem Cluster  $b_x$  und dem nachfolgenden Cluster  $b_y$  und entwickelten eine Wiege-Technik, die die Pixelunterschiede zwischen zwei Clustern analysiert.

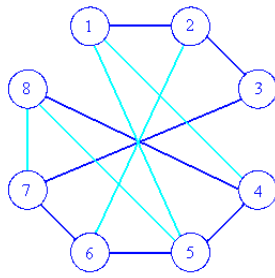


Fig. 6. Hamilton Pfad von 1 nach 8 (dunkelblaue Kanten)

Hierbei ist das Gewicht ( $W_{x,y}$ ) die Summe der Unterschiede der Pixel an der Grenze von Cluster  $b_x$  und Cluster  $b_y$ . [2]

Wenn zwischen jedem Knotenpaar  $b_x$  und  $b_y$  eine Kante mit einem Gewicht ( $W_{x,y}$ ) existiert, ergibt sich eine Adjazenzmatrix eines kompletten Graphen mit  $n$  Knoten, mit Hilfe derer das Problem der Wiederherstellung von fragmentierten Dateien als Graphentheoretisches Problem behandelt werden kann. Nachdem die Gewichte bestimmt wurden, können nun Fragmentpaare identifiziert werden, die im ursprünglichen Dokument adjazent sind [2]:

- Die Wiederherstellung als ein Hamilton Pfad Problem:*  
 Hier wird angenommen, dass sich die Permutation  $P$  der Cluster, die zu einer korrekten Wiederherstellung führt, unter den Permutationen befindet, die die Summe aller Gewichte der in ihr vorkommenden adjazenten Cluster maximiert. Diese Beobachtung führt zu einer Technik, die mit einer hohen Wahrscheinlichkeit die Datei richtig wiederherstellen kann: Es wird die Permutation  $P$  unter allen Permutationen von Grad  $n$  ausgewählt, die insgesamt die größte Summe an Gewichten hat. Die richtige Sequenz  $P$  ist ein Pfad im Graphen, der alle Knoten durchläuft und die Summe der Gewichte der Knoten auf diesem Pfad maximiert. Diesen Pfad zu finden entspricht der Aufgabe einen gewichtsmaximalen Hamiltonschen Pfad in einem vollständigen Graphen zu finden [2]. In der Graphentheorie bezeichnet ein Hamilton Pfad einen Pfad, der jeden Knoten genau einmal passiert (siehe Abbildung 6 auf Seite 7). Shanmugasundaram et al. [4] lösten das Hamilton Pfad Problem mit Alpha-Beta-Pruning der Spieltheorie. Das Prinzip von Alpha-Beta-Pruning: wenn Spielzug  $B$  mit Sicherheit schlechter ist als Spielzug  $A$ , der schon ausgiebig geprüft wurde, lohnt es sich nicht, weitere mögliche Konsequenzen von Spielzug  $B$  zu prüfen [5].
- Die Wiederherstellung als ein  $k$ -Vertex Disjoint Path Problem:*  
 Die Wiederherstellung kann verbessert werden, wenn die Statistiken mehrerer Dateien berücksichtigt werden. Pal et al. entwickelten das Problem zu einem  $k$ -Vertex Disjoint Path Problem [3]. Das Problem der Wiederherstellung ist hier, disjunkte Pfade mit  $k$  Knoten zu finden, bei dem  $k$  gleich der Anzahl der Dateien ist, die mit Hilfe ihres Headers identifiziert wurden. Dies ist ein disjunktes

Pfad-Problem, weil davon ausgegangen wird, dass jeder Cluster auf der Festplatte ausschließlich zu einer Datei gehört. Das  $k$ -Vertex Disjoint Path Problem ist ein NP-schweres Problem. [2]

Im Fall der 24-b BMPs sind die Ausgangspunkte für jede Datei durch ihren Header bekannt. Die Anzahl der Cluster, bzw. Knoten sind ebenfalls bekannt, da die Größe jedes Clusters bekannt ist und die Größe einer BMP Grafik bei einer Untersuchung des Headers festgestellt werden kann. Pal et al. schlugen acht Algorithmen zur Wiederherstellung der Grafiken vor. Die interessantesten Algorithmen unter diesen acht sind die Unique Path (UP) Algorithmen. Eine Up-Algorithmus ist ein Algorithmus, in dem jeder Cluster genau einer Datei zugewiesen wird. Ein UP-Algorithmus ist sinnvoll, da ein Cluster in den meisten Dateisystemen nur zu einer Datei gehört. Allerdings hat ein Up-Algorithmus einen Nachteil: wenn ein Cluster falsch zugeordnet wurde, besteht die Möglichkeit, dass die Datei, zu der dieser falsch zugeordnete Cluster tatsächlich gehört (falls er zu einer Datei gehört), ebenfalls falsch wiederhergestellt wird. D.h. Fehler können kaskadieren. [2]

Die zwei besten der acht UP Algorithmen sind Parallel unique Path (PUP) und Shortest Path First (SPF), die im Folgenden kurz erläutert werden [2]:

#### *Parallel unique Path (PUP)*

PUP ist eine Variation des Dijkstras Single-Source Shortest Path Algorithmus [9], mit dessen Hilfe mehrere Bilder gleichzeitig wiederhergestellt werden können. Beginnend mit dem Header  $b_x$  eines Bildes nimmt der Algorithmus die Cluster mit den besten Übereinstimmungen mit diesen Header  $b_x$  aus der Menge der verfügbaren Cluster. Die Übereinstimmung (das Gewicht) wird aus den Unterschieden der Pixel an der Grenze zwischen zwei Clustern berechnet (siehe oben: Bestimmung der Gewichte im Fall von 24-b Windows BMP Grafiken). Der Cluster mit der besten Übereinstimmung ( $b_y$ ) wird dem Header  $b_x$  des Bildes zugeordnet. Der Cluster  $b_y$  wird anschließend aus der Menge der verfügbaren Cluster genommen (UP-Algorithmus). Nun wird der Cluster mit der besten Übereinstimmung mit Cluster  $b_y$  ermittelt und mit den besten Übereinstimmungen der übrigen Bilder verglichen. Dieser Prozess wiederholt sich bis alle Bilder wiederhergestellt sind.

#### *Shortest Path First (SPF)*

SPF ist ein Algorithmus der annimmt, dass die besten Ergebnisse die geringsten durchschnittlichen Pfad-Kosten haben. Die durchschnittlichen Pfad-Kosten sind die Summe der Gewichte zwischen den Clustern einer wiederhergestellten Datei, geteilt durch die Anzahl der Cluster. Dieser Algorithmus rekonstruiert Bilder nicht gleichzeitig (wie z.B. PUP), sondern in Folge. Nachdem ein Bild wiederhergestellt wurde, werden die durchschnittlichen Pfad-Kosten des Bildes berechnet, aber die Cluster des Bildes werden nicht aus der Menge der verfügbaren Cluster genommen. Die Cluster können

weiterhin verwendet werden um die restlichen Bilder wiederherzustellen. Dieser Prozess wird wiederholt bis alle durchschnittlichen Pfad-Kosten berechnet sind. Anschließend wird das Bild mit den geringsten durchschnittlichen Pfad-Kosten als beste Wiederherstellung gewertet und die Cluster des Bildes werden aus der Menge der verfügbaren Cluster entfernt. Bilder, die einen dieser Cluster enthalten, müssen mit den verbleibenden Clustern neu wiederhergestellt werden. Dieser Prozess wiederholt sich bis alle Bilder wiederhergestellt sind.

Der SPF Algorithmus erreicht mit 88% die beste Präzision, der PUP Algorithmus stellt 83% der Bilder wieder her. Aber der PUP Algorithmus ist schneller und skaliert besser als der SPF Algorithmus. Allerdings skalieren beide Algorithmen nicht gut und das Berechnen von Gewichten zwischen allen vorhandenen Clustern ist bei den heutigen Dateiträgergrößen zu aufwendig.

#### D. SmartCarver

Eine weitere, neuere Carving-Methode ist SmartCarving. Pal et. al. erkannten die Probleme der reinen Graph Theoretic Carving Vorgehensweise mit vorberechneten Gewichten und stellten das SmartCarving Framework zur Verfügung, das fragmentierte Dateien wiederherstellen kann und auch bei großen Datenträgern skaliert [4]. SmartCarving ist eine Kombination von anderen Techniken wie File Structure Based Carving und Graph Theoretic Carving.

Der SmartCarving Prozeß besteht aus drei Hauptkomponenten (siehe Abbildung 7 auf Seite 9):

##### 1) *Preprocessing*

Die Preprocessing-Phase muss auf Laufwerke angewendet werden, die komprimierte oder verschlüsselte Inhalte haben. Windows Vista enthält BitLocker, das, wenn aktiviert, alle Daten auf einem Laufwerk verschlüsselt. Wenn Carving-Techniken verwendet werden sollen, muss der Datenträger entschlüsselt werden. Gleiches gilt, wenn das Gerät komprimierte Daten beinhaltet, diese müssen dekomprimiert werden. Unabhängig von der Verschlüsselungs- oder Kompressionsmethode müssen die Cluster entschlüsselt und dekomprimiert werden, um den Wiederherstellungsprozess fortzuführen. In der Vorlaufphase können auch alle bekannten, d.h. in den Dateisystem Metadaten allokierten Cluster entfernt werden. Dieser Schritt kann übersprungen werden, wenn der Verdacht besteht, dass die Metadaten manipuliert wurden oder beschädigt sind. Es bestehen allerdings große Vorteile bei der Verringerung der Anzahl der Cluster in Bezug auf Geschwindigkeit und Effizienz. [2]

##### 2) *Collating*

Diese Komponente ordnet jedem Cluster ein Dateiformat oder mehrere Dateiformate zu. Dies ist wichtig weil es die Anzahl der Cluster reduziert, die für eine wiederherzustellende Datei infrage kommen. Im Folgenden werden einige bekannte Techniken, die eine solche Zuordnung ermöglichen, kurz erläutert [2]:

- *Keyword/Pattern Matching*

Keyword/Pattern Matching sucht nach festen Sequenzen, z.B. um Header zu erkennen. Ein Cluster der viele `href=` enthält, ist wahrscheinlich ein HTML Cluster. Hiermit können potenzielle Dateiformate erkannt werden. Allerdings müssen zusätzliche Tests folgen.

- *American Standard Code for Information Interchange (ASCII) Character Häufigkeit*

Ist die ASCII Character Häufigkeit in einem Cluster sehr hoch, kann angenommen werden, dass der Cluster zu einem Dateityp gehört, der kein Audio, Video oder Bildformat ist. Auch hier sind nachfolgende Tests nötig.

- *Entropy*

Für jeden Cluster kann eine Entropie (Informationsdichte) berechnet werden um seinen Dateityp einzugrenzen. Nachfolgende Tests sind weiterhin nötig.

- *File Fingerprints*

File Fingerprints sind eine der ersten robusten Techniken, die Dateitypen bestimmen können. Ein File Fingerprint wird mit Byte-Mustern der einzelnen Cluster verglichen. Ein File Fingerprint ist z.B. eine durchschnittliche Häufigkeitsverteilung (BFD: byte frequency distribution) von 256-Byte Werten oder die durchschnittliche Häufigkeit zwischen Byte-Wert-Paaren (BFC: byte frequency cross-correlation) in einer Datei. Dies alleine reicht aber noch nicht für die Identifikation eines Clusters aus, da nicht jeder Cluster einen Header und Footer (die ebenfalls analysiert werden müssen) beinhaltet. Auch bei einer verbesserten Technik, die eine Menge von Modellen für jeden Dateityp definiert, anstatt einem einzigen Modell (byte frequency analysis (BFA)), wird die Präzision umso schlechter, je kleiner die Teilmengen sind. Da Cluster i.d.R nicht größer als 4.096 Bytes sind, wird die Präzision eines Tools weiter abnehmen, wenn File Fingerprints verwendet werden.

- *Klassifizierung für File Carving*

Hier wird die Rate of Change (RoC) in den Byte Werten berechnet. Der RoC ist die absolute Differenz zwischen zwei aufeinanderfolgenden Bytes in einem Cluster. Diese Methode erreicht beim identifizieren von JPEG Clustern eine Präzisionsrate von 99%.

##### 3) *Reassembly*

In dieser Phase wird zunächst nach einem Fragmentierungspunkt einer wiederherzustellenden Datei und anschließend nach dem Anfangspunkt des nächsten Fragments der Datei gesucht. Eine Datei mit k-Fragmenten besitzt k-1 Fragmentierungspunkte. Es gibt mehrere Möglichkeiten herauszufinden, ob Cluster y in einer Datei direkt nach Cluster x kommt:

- *Keyword/Dictionary*

Eine einfache Technik für das Zusammenführen von Clustern ist, eine Liste von Keywords oder ein Standard-Wörterbuch zu verwenden um zu ermitteln, ob zwei Cluster zusammengeführt werden sollten. Eine Wörterbuch-basierte Zusammenführung entsteht, wenn ein bekanntes Wort über die Cluster-Grenzen liegt.

- *File Structure Merging* (siehe auch *File Structure Based Carving in Kapitel III-A*)

Bei bestimmten Dateitypen kann die Kenntnis der Struktur eines Dateityps zu einer Zusammenführung mehrerer Cluster führen. Dies kann z.B. der Fall sein, wenn in einem Dateityp die Länge und ein CRC (Cyclic Redundancy Check) der Datei angegeben wird (z.B. PNG).

- *SHT-PUP*

SHT-PUP ist ein abgeänderter PUP Algorithmus bei dem angenommen wird, dass für jedem Cluster, der zu einem Pfad hinzugefügt wird, die Wahrscheinlichkeit hoch ist, dass die anschließenden Cluster ebenfalls zu diesem Pfad gehören. Der Algorithmus ermittelt mit Hilfe eines sequenziellen Hypothesentests, ob aufeinanderfolgende Cluster zusammengelegt werden sollten. Der Algorithmus ermittelt dies nicht nur auf die Struktur des Dateityps, sondern auch auf die Inhalte der einzelnen Dateien bezogen. Dieser modifizierte Algorithmus nennt sich Sequential Hypothesis-Parallel Unique Path (SHT-PUP).

Eine der ersten Umsetzungen von SmartCarving war von Nick Mikus in Foremost. Foremost lieferte mit SmartCarving weit bessere Ergebnisse als zuvor mit File Structure Based Carving. In 2006 veranstaltete Digital Forensic Workshop (DFRWS) einen Wettbewerb, in dem fragmentierte Dateien aus einem randomisierten Datensatz wiederhergestellt werden sollten. Joachim Metz und Robert-Jan Mora entwickelten daraufhin ein Tool namens ReviveIt (Revit, das später umbenannt wurde revit06), das ein Machbarkeitsbeweis für SmartCarving ist. Auch Revit06 lieferte weit bessere Ergebnisse als Carver, die Header/Footer und Header/Maximum (Datei) Größe Carving Methoden verwendeten. [19]

Ein weiteres Tool, das SmartCarving verwendet ist Adroit Photo Forensics, das bekannte Bildformate wiederherstellt. Adroit Photo Forensics unterstützt u.a. die Wiederherstellung von fragmentierten Dateien mit SmartCarving und Guided-Carving.

#### E. In-Place Carving/ Zero-Storage Carving

Da Carving Tools viele False-Positives generieren, benötigen sie viel Speicherplatz für ihre Ergebnisse. Mithilfe von In-Place Carving ist Carving auch mit wenig zur Verfügung stehendem Speicherplatz möglich. Statt die wiederhergestellten Dateien auf einen Datenträger zu kopieren, legt das In-Place Carving eine Datenbank an, in der die Speicherorte der gefundenen Dateien gespeichert werden. Die Dateien können anschließend auf Anfrage direkt aus dem Image zusammengesetzt werden. Dies bedeutet, dass auch nur mit einem Memory-

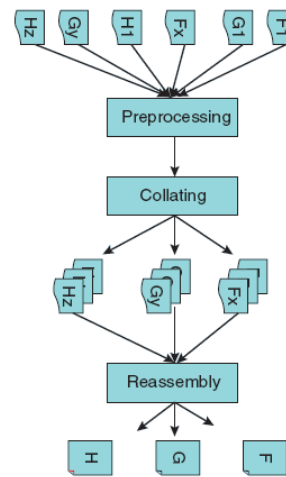


Fig. 7. SmartCarver [2]

stick und einen Laptop, große Ziele untersucht werden können. In-place-Carving ist besonders nützlich, wenn eine schnelle Datenrettung bei Zielen durchgeführt werden muss, bei denen nur wenig zusätzlicher Speicherplatz zur Verfügung steht. [13]

#### F. File trimming

Beim File trimming werden nicht zu einer Datei gehörende Daten am Ende der Datei entfernt. Hierbei gibt es zwei Möglichkeiten [28]:

- 1) *es ist ein intakter Footer vorhanden*  
Falls ein intakter Footer vorhanden ist, kann die Datei nach dem Footer gekürzt werden.
- 2) *es ist kein Footer vorhanden*  
Bei Dateitypen ohne Footer wird die Datei solange gekürzt, bis sie nicht mehr validiert. Anschließend wird das zuletzt gekürzte Byte wieder hinzugefügt.

#### G. Validierung

Es gibt viele Möglichkeiten zu überprüfen, ob eine Datei gültig ist. Carving Tools, die keine umfangreiche Validierung der Dateien vornehmen, produzieren viele False-Positives. Im Rahmen einer Validierung wird geprüft, ob eine Datei fehlerfrei wiederhergestellt wurde. Ein Validierer stellt fest, ob eine Datei die Regeln des spezifischen Dateityps befolgt oder verletzt. Bekannte Validierungsmethoden sind [28]:

- *Validierung von Container-Strukturen*  
Bei der Validierung von Container-Strukturen werden Komponenten innerhalb einer Datei validiert, zum Beispiel ein JPEG-Bild innerhalb einer Word-Datei. Beispiel: An einer bestimmten Stelle in der Container Datei, z.B. einer Word-Datei, befindet sich laut Dateiformat ein Integer. Dieser Integer stellt einen Zeiger dar, der auf ein anderes Objekt innerhalb der Container-Datei zeigt. In diesem Fall kann überprüft werden, ob der Integer in einem vordefinierten Bereich liegt und wenn ja, ob der Zeiger auf ein anderes gültiges Objekt zeigt.
- *Validierung mit Dekompression*

Im Rahmen einer Validierung mit Dekompression wird der Inhalt einer Datei validiert. Zum Beispiel wird in einem Microsoft Office Dokument überprüft ob alle Charaktere in der Text Section gültig sind, oder in einem JPEG ob alle Huffman Symbole gültig sind.

- **Semantische Validierung**  
Semantische Validierung wird eingesetzt, um Informationen eine Bedeutung zu geben und damit Lücken und Unstimmigkeiten in den vorhandenen Daten zu finden. Beispiel: Die Buchstaben hospi sind die letzten Buchstaben eines Sektors und die Buchstaben tals sind die ersten drei Buchstaben des nachfolgenden Sektors. Weiterhin ist bekannt, dass das Dokument von Gesundheitspolitik handelt. In diesem Fall ist es sehr wahrscheinlich, dass die Sektoren zur selben Datei gehören.
- **Manuelle Validierung**  
Wenn das Carving Tool eine Folge von Bytes findet, die den gewünschten Anforderungen entspricht, werden die Bytes in eine Datei, die anschließend manuell geöffnet und geprüft wird, gespeichert. False-Positives müssen manuell herausfiltert und Dateien evtl. nachbearbeitet werden. Eine unterstützte manuelle Bearbeitung ist mit Guided Carving möglich (siehe Kapitel III-H).

**H. Guided Carving**

Guided Carving ist eine File Carving-Technik, die von Adroit Photo Forensics (APF) eingeführt wurde um fragmentierte Dateien wiederherzustellen. APF ist ein kommerzielles Forensic Software-Paket von Digital Assembly. Es ist auf die Verwertung und Analyse von digitalen Fotos spezialisiert. Mithilfe von Guided Carving kann ein Benutzer versuchen, eine fragmentierte Datei, die nicht vollständig wiederhergestellt wurde, wiederherzustellen.[8] Der Guided Carving Prozess [8]:

- 1) Der Benutzer öffnet ein Foto, das nicht richtig wiederhergestellt wurde (siehe Abbildung 8 auf Seite 10).
- 2) Er identifiziert den ersten Fehler auf einem Foto, indem er auf den Fehler klickt
- 3) Adroit Photo Forensics präsentiert ihm anschließend die bestmöglichen Block-Treffer (siehe Abbildung 9 auf Seite 10).
- 4) Der Benutzer geht durch die ihm angebotenen Blöcke um den korrekten Fortsetzungsblock zu finden.
- 5) Er kann anschließend wählen ob er den Rest der Datei manuell erstellen will oder ob SmartCarving übernehmen soll.

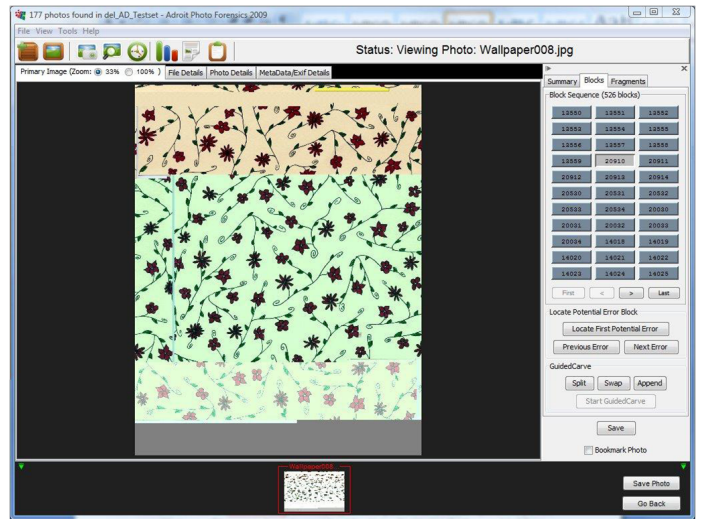


Fig. 8. Adroit Photo Forensic [8]

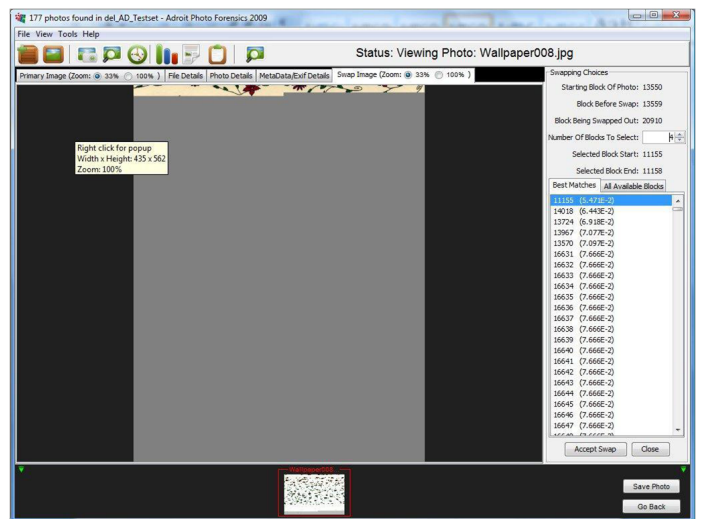


Fig. 9. Adroit Photo Forensic [8]

**IV. SEMANTISCHE ANALYSE IN DER DIGITALEN FORENSIK**

**W**ÄHREND sich Carving Methoden auf die Wiederherstellung gelöschter Daten konzentrieren, geht es in der semantischen Analyse hauptsächlich darum, sehr große Datenmengen derart zu analysieren, sodass diese interpretiert werden können, um daraus wiederum Schlüsse (implizite Informationen) über die Daten zu erheben. Der Begriff der semantischen Analyse ist dabei nicht konkret definiert. Vielmehr kann darunter eine Ansammlung verschiedener Konzepte verstanden werden, um dadurch digitale Spuren (bzw. Beweise) auffindig zu machen, die vor Gericht verwendet werden können. Der Prozess der semantischen Analyse ist sehr umfangreich und erfordert ein tiefgehendes Verständnis der eingesetzten Konzepte, welche in dieser Ausarbeitung unter anderem behandelt werden. Zunächst jedoch erfolgt eine Erläuterung grundlegender Begriffe, um die Idee der semantischen Analyse zu verdeutlichen.

### A. Unterscheidung von Informationen und Daten

Damit beide Begriffe nicht synonym verwendet werden, wird an dieser Stelle eine Abgrenzung definiert. Nach Rich Smith [35] besteht ein formaler und praktischer Unterschied zwischen Informationen und Daten. Während Daten als logisch gruppierte Informationseinheiten angesehen werden, sind Informationen eine subjektive Interpretation von Daten. Die semantische Analyse beschäftigt sich hier mit beiden Begriffen, wobei zuerst Daten analysiert werden, um diese anschließend mit maschineller Hilfe zu interpretieren.

### B. Digitale Spuren

Digitale Spuren ("digital evidence") sind alle Spuren, die auf Daten basieren, welche in Computersystemen gespeichert oder übertragen worden sind [38]. Dabei sind diese zunächst nur physische Spuren:

- Magnetisierung auf der Oberfläche einer Festplatte
- Elektromagnetische Wellen auf einem Datenkabel
- Ladezustand von Transistoren im Hauptspeicher

Für die meisten Aufschlüsse reicht es jedoch aus, nur die "Abstraktion" (die digitalen Werte) der physischen Spuren anzuschauen. Generell existiert zwischen digitalen Spuren und deren Abstraktion eine aufeinanderbauende Abhängigkeitskette:

- Digitale Spuren benötigen Werkzeuge zur Aufbereitung
- Werkzeuge zeigen nur eine Abstraktion, bzw. Interpretation der physischen Spuren
- Abstraktionen können mehrere Abstraktionsebenen enthalten
- Jede Abstraktionsebene kann wiederum verschiedene Interpretationen beinhalten

Zur Verdeutlichung: Eine E-Mail Datei kann beispielsweise wie folgt interpretiert werden:

- Interpretation der Magnetisierung der Festplatte (Bits)
- Interpretation der Bits durch eine Zeichencodierung
- Interpretation der Zeichen durch ein Dateisystem
- Interpretation der Daten im Dateisystem als zusammengehörige Datei
- Interpretation der Datei als E-Mail
- Interpretation der Semantik des Textes in der E-Mail

Fokus der semantischen Analyse ist insbesondere die letztgenannte Abstraktionsebene.

### V. HERAUSFORDERUNG DER SEMANTISCHEN ANALYSE

Während der forensischen Analyse fallen generell sehr große Datenmengen aus unterschiedlichen Dateien an, z.B.:

- Dokumente / User-generierte Dateien (PDF's, Doc's, Excel Sheets, Powerpoint Folien, etc.)
- Konversationen via E-Mails / Instant Messaging Clients (Skype-Anrufen, MSN/ICQ Chats, etc.)
- Systemevent Log's (Netzwerk, Dienste, etc.)
- Web-Content (Blogs, Messaging in Social Networks, etc.)
- Mobile-Content (Kontaktlisten, SMS, Termine, Textnotizen, etc.)

Als Quellen wiederum wären unter anderem zu nennen:

- Festplatten (intern/extern)
- Speichermedien (CD, DVD, Blu-ray, Magnetbänder, etc.)
- Speicherkarten (Compact Flash, Secure Digital, Smart-Media Cards, etc.)
- Mobile Speichergeräte (MP3 Player, USB-Sticks, Handy, PDA, etc.)
- Netzwerk

Unglücklicherweise teilen sich die meisten aller Daten eine gemeinsame Eigenschaft. Sie lassen sich nicht ohne weiteres maschinell verarbeiten, bzw. interpretieren. Dies hat zur Folge, dass Ermittler mit herkömmlichen Suchanfragen nicht in der Lage sind, relevante Informationen aus den Datenmengen zu finden. Diese Problematik ist damit begründet, dass die meisten Daten keinerlei Struktur aufweisen (gegenüber beispielsweise Datenbanken), um dadurch gezielt verarbeitet zu werden. Viele forensische Werkzeuge, welche auf diese Daten operieren, haben ebenfalls keine einheitliche formale Repräsentation und folgen keine Standards. Hinzu kommt noch, dass die meisten dieser Werkzeuge sich proprietärer Software bedienen, welche von forensischen Experten weder analysiert noch verifiziert werden können.

Bis heute existiert kein offenes Datenformat für digitale Beweise, bzw. keine generelle Prozedur für die digitale forensische Analyse [37]. Da sich letzteres nur schwer realisieren ließ, entstand seitens der forensischen Forschung der Wunsch, ein standardisiertes Format zu finden, welches den Austausch digitaler Beweise zwischen verschiedenen Systemen ermöglichen soll. Forscher der "DFRWS CDESf Working Group" [41] haben im Jahre 2005 versucht, dass *Common Digital Evidence Storage Format* (CDESf) durchzusetzen. Unter anderem sollte das CDESf Format ein bitweises Image einer Festplatte beinhalten, die Meta-Information, wo und wie das Image erstellt wurde, ein digitales Foto der Festplatte und den Namen der Person, welches das Image erstellt hatte sowie der verwendeten Identifikations-Nummer des Falles. Der Versuch der Standardisierung dieses austauschbaren Formats scheiterte leider aufgrund der Tatsache, dass sich die Forschungsgruppe im August 2007 (hauptsächlich wegen Mangel an Ressourcen) aufgelöst hatte [39], daher besteht nach wie vor der Bedarf eines standardisierten Formats für digitale Beweise.

Für gewöhnlich besteht in der forensischen Analyse der

allererster Schritt darin, ein Image des jeweiligen Mediums zu erstellen, welches untersucht werden soll. Ein derartiges Image wird in der Literatur [32] auch als *binary data image blob* bezeichnet. Das Erstellen von Images zählt zu den zentralen Aufgaben der Host Forensik. Neben Images existieren weitere digitale Beweismittel, wie beispielsweise Register-, Memory- oder Chache-Dumps. Derartige Spuren müssen zunächst extrahiert und in eine lesbare Form übersetzt werden, dies geschieht in der Regel durch forensische de-facto Standardwerkzeuge (z.B. "EnCase", "PyFlag" oder "FTK"). Die daraus resultierenden (lesbaren) Daten können anschließend für die semantischen Analyse verwendet werden.

## VI. KONZEPTE DER SEMANTISCHEN ANALYSE

Ausgehend von dem zuvor erstelltem Image, geht es im nächsten Schritt darum, relevante Informationen innerhalb des Images ausfindig zu machen. Um dieses Ziel zu erreichen, werden verschiedene Konzepte und Methoden aus den folgenden Disziplinen benötigt:

- Text Mining
- Information Retrieval
- Maschinelles Lernen
- Künstliche Intelligenz
- Computerlinguistik (hier insbesondere die Teilgebiete: forensische Linguistik als auch allgemein: Natural Language Processing)

Diese Disziplinen haben sich weltweit für unterschiedliche Anwendungszwecke im Bezug auf Erschließung und Verarbeitung vom expliziten, bzw. impliziten Wissen bewährt (mitunter auch auf dem forensischen Gebiet). In den folgenden Unterabschnitten werden einige fundamentale Konzepte der genannten Disziplinen näher erläutert, ohne die eine semantische Analyse praktisch undurchführbar wäre.

### A. Text Mining

In der Literatur [49], [50] gibt es für den Begriff des Text Minings eine Vielzahl von Definitionen, die sich hauptsächlich in zwei Gruppen einteilen lassen:

- Text Mining umfasst alle Techniken, die es dem Benutzer ermöglichen, interessante Informationen aus externen Ressourcen zu extrahieren ([50] - R. D. Hackathorn).
- Text Mining umfasst alle Techniken zum Entdecken und automatischen Extrahieren von neuen, zuvor unbekanntem Informationen aus Texten ([49] - M. A. Hearst).

Die zweite Definition ist eine Eingrenzung der ersten Definition, bezüglich der Art der zu entdeckenden Informationen (neue, zuvor unbekanntem Informationen) und der Art der Informationsquellen (unstrukturierte Texte). Ein möglicher Text Mining Prozess ist wie folgt durch [48] gegeben:

- 1) **Aufgabendefinition:** Festlegung der Problemstellung und Ableiten der Text-Mining-Ziele.
- 2) **Dokumentselektion:** Ausgehend von der Aufgabendefinition werden die potenziell relevanten Dokumententexte identifiziert.
- 3) **Dokumentaufbereitung:** Da die Texte in unstrukturierter Form vorliegen, ist eine Extraktion der relevanten Informationen aus dem Text notwendig. Hierfür werden bestimmte Terme aus dem Text extrahiert, die in den anschließenden Schritten den Text repräsentieren. Ein Term kann dabei z.B. ein Wort, ein Wortstamm, aber auch mehrere zusammengesetzte Wörter sein. Der Text wird dadurch in eine maschinenlesbare Form gebracht.
- 4) **Text Mining Methoden:** Hier werden Verfahren aus dem Data Mining auf die nun strukturierten Texte angewandt. Die Dokumententexte können zueinander in Beziehung gesetzt und Themen zugeordnet werden.
- 5) **Interpretation und Evaluation der Ergebnisse:** Ausfiltern und Bewertung handlungsrelevanter Text-Mining-Ergebnisse.
- 6) **Anwendung der Ergebnisse:** Das aus den Texten gewonnene Wissen kann nun auf die in der Aufgabenstellung definierte Problemstellung angewandt werden.

Text Mining Methoden werden häufig dann angewendet, falls die zugrundeliegenden (Text-)Daten schwach- oder völlig unstrukturiert vorliegen, was in der Praxis nicht selten der Fall ist. Ohne den Einsatz von Text Mining ist eine semantische Analyse praktisch undurchführbar, da diese fundamentale Aufgaben beinhaltet, die das Aufspüren von relevanten Informationen erst ermöglicht.

### B. Information Retrieval

Information Retrieval (IR) ist ursprünglich ein angewandtes Konzept aus der Disziplin des Web Minings, kommt jedoch auch beim Text Mining zum Einsatz (vergleiche Schritt A.2). IR verfolgt das Ziel, eine schnelle, inhaltsorientierte und "unscharfe" Suche in den unstrukturierten Datenmengen zu ermöglichen. Dazu werden Daten ausfindig gemacht, die mit einer gewissen Wahrscheinlichkeit für ein gegebenes Szenario relevant sein können. Dabei vertrauen IR Methoden auf statistische Modelle und Annahmen, ohne wirklich die Semantik der Daten zu verstehen.

Das Resultat eines IR Prozesses ähnelt dem Ergebnis einer Anfrage bei einer Suchmaschine. Es werden damit also sämtliche Ergebnisse zurückgeliefert, die zu der gegebenen Anfrage (Query) passen. In der Regel werden diese zusätzlich nach Relevanz sortiert, sodass wichtige Informationen an erster Stelle platziert werden und unwichtigere wiederum weiter hinten in der Reihenfolge zu finden sind. Die Sortierung nach Relevanz erfolgt typischerweise durch sogenannte Page-Rank Algorithmen, die ebenfalls in der Disziplin des Web Minings ihren Ursprung haben. Ungeachtet dessen, beinhalten die zurückgelieferten Ergebnisse noch zuviel Noise (überflüssige Informationen), sodass der Benutzer hier eine Verfeinerung der

gefundenen Informationen von Hand vornehmen muss. Diesen Aufwand zu umgehen ist wiederum eines der Aufgaben des Information Extraction Konzepts.

C. Information Extraction

Genauso wie Information Retrieval beschäftigt sich Information Extraction (IE) mit der Suche nach relevanten Informationen in großen Textmengen. IE jedoch geht dabei einen Schritt weiter. Während sich IR damit beschäftigt, nur die relevanten Textdokumente zu finden, geht es bei IE darum bestimmte Informationen aus den relevanten Dokumenten zu extrahieren (vergleiche Schritt A.3). Die Ausgangsdaten für den IE-Prozess sind hauptsächlich in Form von unstrukturierten oder halbstrukturierten Dokumenten gegeben. Die Aufgabe eines IE-Systems ist es, bestimmte Fakten, die in den Ausgangsdaten enthalten sind, aufzufinden und in eine vorgegebene strukturierte Form (ein sogenanntes "Template"), zu überführen. Für die semantische Analyse stellt das resultierende Template jedoch nur eine Zwischenlösung dar. Die Gründe hierfür sind zum einem, dass das IE System das vollkommene Verständnis des Textes nicht garantieren kann und zum anderem, dass nur Fakten extrahiert werden, die zu der vorgegebenen Wissensdomäne (z.B. "Betrug", "Raub", "Erpresung", et.) thematisch passen. Die folgende Illustration verdeutlicht ein IE System:

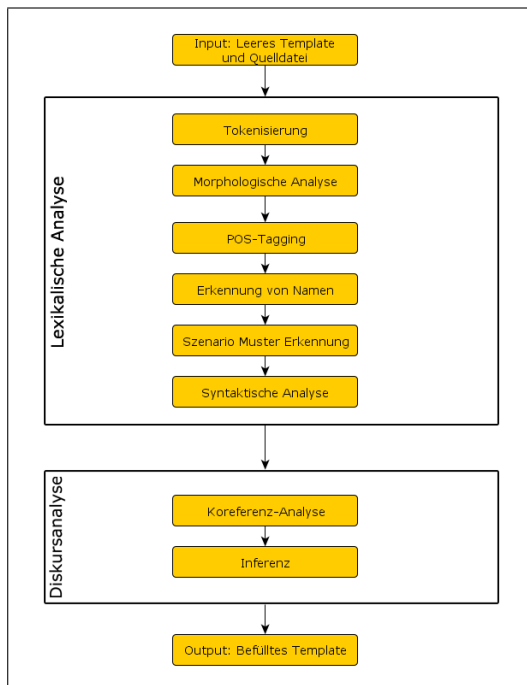


Fig. 10. Information Extraction Prozess

Erläuterung der einzelnen Phasen:

- **Tokenisierung:** Der textuelle Eingabestrom wird in Sätze und einzelne Wörter zerlegt.
- **Morphologische Analyse:** Die einzelnen Wörter werden analysiert, um verschiedene Wortformen zu erkennen, die

normalerweise durch Flexion entstehen. Das sind Wörter, die sich von ihrem Stamm syntaktisch unterscheiden, aber semantisch die gleiche Bedeutung haben. Anschließend werden zusammengesetzte Wörter in ihre Einzelteile zerlegt.

- **Part-Of-Speech Tagging:** Die Wörter im Text werden mit der entsprechenden Wortart markiert, z.B.: Verb, Nomen, Adjektiv etc.
- **Erkennung von Namen:** Mit Hilfe dieser Technik werden verschiedene Namen (Personen, Firmen, Orte) und andere Elemente wie Währungen, Datum und Zeitangaben erkannt und entsprechend markiert.
- **Szenario Muster Erkennung:** Bei diesem Schritt werden die Ereignisse und die Beziehungen der Fakten, die für das IE-System relevant sind, erkannt.
- **Syntaktische Analyse:** Bei der Extrahierung von Fakten können Informationen über die syntaktische Struktur der Sätze von Vorteil sein. Besonders Nomen können dazu beitragen, die als Hauptwörter in einem Satz angesehen werden können. Zusätzlich können durch die Kennzeichnung von verbalen Phrasen funktionale Relationen zwischen ihnen erkannt werden.
- **Koreferenz-Analyse:** Koreferenz ist ein Phänomen, welches auftritt, wenn auf sprachlicher Ebene mit verschiedenen Formen auf dasselbe Objekt ausserhalb eines Texts referenziert wird. Bei der Koreferenz-Analyse geht es darum mithilfe von Klassifizierern (dazu später mehr) entscheiden zu können, ob zwischen zwei Phrasen, eine Koreferenz besteht oder nicht.
- **Inferenz:** Bei diesem Schritt wird aus bereits bekanntem Wissen, neues (implizites) Wissen abgeleitet.

Wie aus der Abbildung, bzw. dem Prozess entnommen werden kann liegt die Hauptproblematik des IE Konzeptes darin, dass ausschließlich nur textuelle Dateien als Eingabedaten in Frage kommen. Nicht-textuelle Daten hingegen, benötigen andere Verfahren um maschinell verarbeitet, bzw. interpretiert zu werden. Im nächsten Abschnitt werden Möglichkeiten erläutert, um diese Problematik entgegenzuwirken.

D. Annotation nicht-textueller Daten

In der semantischen Analyse werden sehr oft textuelle Daten (z.B. E-Mails, Dokumente, Logdateien) untersucht. Dies liegt an der Tatsache, dass die meisten Informationen in geschriebener Form vorliegen. Leider ist es aber auch oft der Fall, dass relevante Informationen in Bildern, Audio- oder Videodateien enthalten sind. Es stellt sich daher die Frage, wie derartige Dateien für die Analyse verwendet werden können.

Hierfür existieren spezielle Werkzeuge (sogenannte "Annotatoren"), dessen Zielsetzung es ist, Ermittlern die Möglichkeit zu geben, Daten die keinerlei textuelle Informationen beinhalten (hauptsächlich Multimedia-Formate) mit Metadaten anzureichern. Solche Daten beinhalten (z.B. bei Bild Dateien) Informationen wie: beteiligte Objekte

/ Subjekte auf dem Bild, Zeitstempel (falls vorhanden, ansonsten Schätzungen anhand der Lichtverhältnisse), Gegebenheiten wie z.B. Ort der Aufnahme innerhalb des Bildes, etc. Ist die Annotation erst abgeschlossen, kann begonnen werden nach verschiedenen Mustern und Phänomene in den Daten zu suchen. Solche Phänomene können z.B. versteckte Hinweise beinhalten, die erst mit einer semantischen Analyse aufgedeckt werden können, dabei können auch die zuvor genannten Text Mining Methoden zum Einsatz kommen.

Aufgrund des immensen Aufwandes ist die Annotation der Metadaten nicht für jede forensische Analyse sinnvoll. Geht es jedoch um sensible Fälle, wie z.B. Mord, Terrorismus oder schwere Wirtschaftsvergehen (z.B. Steuerhinterziehung in Millionenhöhe), dann kann der Einsatz solcher Werkzeuge durchaus in Betracht gezogen werden. Beispiele für Werkzeuge zur Annotation, bzw. auch Transkription ("Übersetzung" sprachlicher Ausdrücke von einem System in ein anderes) wären unter anderem:

- **EXMARaLDA:** Steht für "Extensible Markup Language for Discourse Annotation". Es handelt sich hierbei um ein System von Konzepten, Datenformaten und Werkzeugen für die computergestützte Transkription und Annotation gesprochener Sprache, sowie für das Erstellen und Auswerten von Korpora gesprochener Sprache. EXMARaLDA wird im Teilprojekt "Computergestützte Erfassungs- und Analysemethoden multilingualer Daten" des Sonderforschungsbereichs "Mehrsprachigkeit" (SFB 538) der Universität Hamburg entwickelt [46].
- **Transana:** Hierbei handelt es sich um ein Transkriptions- und Analyseprogramm zur Verarbeitung von Videodaten (z.B. avi, mpg, wmv, etc.) Das Programm, welches am Wisconsin Center for Education Research der University of Wisconsin-Madison entwickelt wurde, eignet sich vor allem bei der Annotation größerer Datenmengen [47].

Neben den obengenannten Disziplinen entstanden in den letzten Jahren viele verwandte Konzepte, welche versuchen Informationen von vorne rein derart zu modellieren, sodass sich diese, ähnlich wie Informationen aus einer Datenbank, abfragen lassen können.

Ein herausragendes Konzept, welches sich am meisten damit beschäftigt, ist der, der sogenannten "Ontologien" (auf die im nächsten Abschnitt näher eingegangen wird). Dieses Konzept gewinnt heutzutage dank der umfangreichen Möglichkeiten, in nahezu jeden Bereich, immer mehr an Bedeutung, unter anderem auch in der forensischen Analyse.

## VII. EINFÜHRUNG IN ONTOLOGIEN

In diesem Abschnitt wird ein kurzer Überblick darüber gegeben, was Ontologien sind und in wie weit diese in der digitalen Forensik verwendet werden können.

### A. Begriffserklärung

Der Begriff Ontologie stammt als solches aus dem Griechischen und setzt sich aus den Wörtern "on" (dem Genitiv von "Sein") sowie "logos" ("Lehre von"), zusammen. Der Literatur zufolge nach [33], hat die Ontologie ihren Ursprung in der Metaphysik (Teilgebiet der Philosophie) und befasst sich in diesem Zusammenhang mit der Frage, warum "etwas" existiert.

Im Kontext der Informationstechnologie, wurde der Begriff vor fast drei Jahrzehnte von Forschern auf dem Gebiet der künstlichen Intelligenz, für die Modellierung von Wissen aufgegriffen. Allgemein sind Ontologien im informationswissenschaftlichen Sinne seitdem als formale Beschreibungen von Konzepten innerhalb einer Wissensdomäne definiert. Ontologien sorgen also für ein gemeinsames Verständnis dieser Konzepte. Die Intention war es, Wissensdomänen in einer geeigneten logischen, maschinenlesbaren Form zu modellieren, um so eine Schnittstelle für agentenbasierte Softwaresysteme zu haben, die auf dieses Wissen zurückgreifen, um Aufgaben mit Hilfe von automatisiertem Schließen zu lösen. Dabei soll implizites Wissen so weit wie möglich externalisiert werden, sodass widersprüchliche Interpretationen vermieden werden können.

Häufig werden die Begriffe "Ontologien" und "Semantic Web" synonym gebraucht, dies jedoch ist grundlegend falsch. Richtig dagegen ist, dass Ontologien eine zentrale Rolle im Semantic Web einnehmen. Das Semantic Web ist dabei der Oberbegriff für zahlreiche Standards, Techniken und Ideen zur Abfrage, Übertragung, Speicherung und Verarbeitung von Daten, welche der maschinenlesbaren Hinterlegung von Sinnesinformationen dienen [45].

### B. Bestandteile von Ontologien

Ontologien bestehen für gewöhnlich aus den folgenden Komponenten:

- **Konzepte:** Objekte, ähnlich wie in Programmiersprachen, Beispiele: *Fahrzeug, Lebewesen, Gebäude, etc.*
- **Individuen/Instanzen:** Instanz eines Konzeptes, Beispiel: *Opel, Bullterrier, Villa, etc.*
- **Eigenschaften/Relationen:** z.B. *ist\_Ein, lebt\_In, befreundet\_Mit, etc.*
- **Axiome:** Aussagen über Begriffe, z.B. *Ein Mensch ist entweder ein männliches oder weibliches Lebewesen*

Mithilfe dieser Komponenten können komplexe Strukturen für eine festgelegte Domäne erstellt werden, wobei einzelne Domänen wiederum miteinander vernetzt werden können. Die Möglichkeit einer solchen Zusammenführung, ist eines der zentralen Vorteile von Ontologien (die nicht immer leicht zu realisieren ist). Die folgende Illustration verdeutlicht anhand einer (kleinen) Ontologie, inwieweit die genannten Komponenten zusammenhängen:

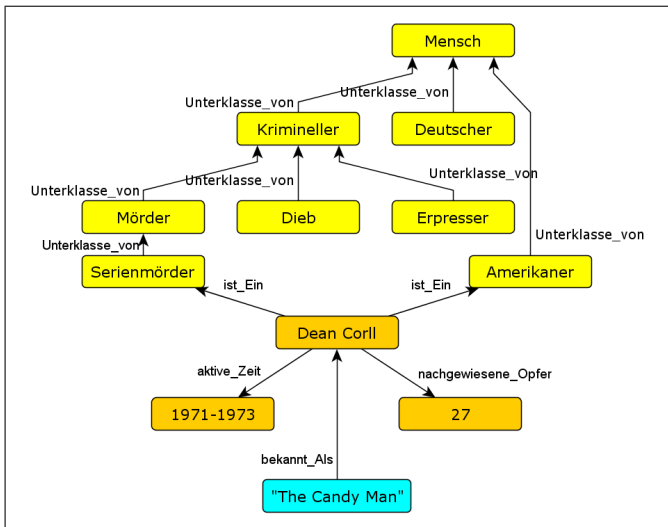


Fig. 11. Ausschnitt einer Ontologie aus der Domäne der Kriminologie

Die gelb markierten Knoten in der Abbildung repräsentieren die Konzepte und damit die inhaltliche Struktur der Ontologie. Die orange gefärbten Knoten wiederum, stellen die Instanzen (Objekte der realen Welt) dar, die sich durch ihre Identität von den "abstrakten" Konzepten unterscheiden. Der türkis gefärbte Knoten ist ein Axiom (im konkreten Fall ein Synonym für den Namen des Serienmörders *Dean Corll*).

Anhand des Beispiels lässt es sich erahnen, welche Möglichkeiten es gibt, forensische Szenarien zu modellieren. Zu erwähnen sei an dieser Stelle, dass es nicht faktisch belegt werden kann, ob staatliche Institutionen (z.B. BND, FBI, CIA, etc.) Ontologien für reale Szenarien praktisch umsetzen.

*C. Anwendungen von Ontologien in der Forensik*

Mittels Ontologien können typische Fragestellungen, die innerhalb eines forensischen Analyseprozesses aufkommen beantwortet werden, beispielsweise:

- Wer hatte offiziell Zugriff auf die Daten?
- Welche Dateien und Informationen wurden von wem und wann zuletzt genutzt?
- Von wem wurden Daten wohin (USB-Stick, CD/DVD, externe Festplatte, etc.) kopiert?
- Gab es unauthorisierte Zugriffe? (Hacker von Außen und/oder Innen, Passwort Cracking Software)
- Welche Internetseiten wurden besucht?

Um eine automatische Beantwortung dieser Fragen zu ermöglichen müssen diese maschinell übersetzt und zusätzlich das zugrundeliegende Wissen als Ontologie erfasst werden. Hierbei muss jedoch zunächst geklärt werden, wie Ontologien überhaupt realisiert werden können.

*D. Realisierung von Ontologien*

Ontologien werden mit Hilfe sogenannter Wissensrepräsentationssprachen ausgedrückt. Es gibt eine Vielzahl solcher

Beschreibungssprachen, beispielsweise RDF oder OWL ("Web Ontology Language"), auf die im nächsten Abschnitt genauer eingegangen wird. Prinzipiell existieren zunächst drei Möglichkeiten, um Ontologien zu erstellen:

- **Manuell:** Die Ontologie wird hier ausschließlich händisch generiert. Dies hat eine hohe Qualität zur Folge, erfordert jedoch gleichzeitig einen großen Arbeitsaufwand, was somit nur bei kleineren Ontologien sinnvoll ist. Um die mühevollen Erstellung zu vereinfachen, können unter anderem Ontologie-Editoren eingesetzt werden. So hat sich hier beispielsweise der Editor "Protégé" bewährt, welches an der Stanford University in Kalifornien (USA) entwickelt wurde. Dieser erlaubt es Ontologien mithilfe einer durchdachten GUI deutlich schneller zu gestalten anstatt diese explizit zu kodieren. Trotz einer solchen Werkzeugunterstützung ist das manuelle Erstellen von Ontologien jedoch langwierig, aufwendig und kostspielig. Die manuelle Erstellung einer nützlichen Ontologie mit mehreren hunderttausend Einträgen, ist daher oft wirtschaftlich inpraktikabel und unzumutbar (diese Problematik ist bekannt unter *knowledge acquisition bottleneck*).
- **Semi-automatisch:** Bei diesem Ansatz wird versucht, die Ontologieerstellung zumindest teilweise zu automatisieren, um den Aufwand bei rein manueller Erstellung entgegenzuwirken. Ermöglicht werden kann dies indem beispielsweise effektivere Arbeitsumgebungen für Ontologieentwickler oder gar teilautomatisierte Tools, mit denen ein Domänenexperte ohne Entwicklererfahrung sein Wissen in einer Ontologie formulieren kann, geschaffen werden [34].
- **Voll-automatisch:** Hier wird anhand von Lernprozeduren versucht, Teile der Ontologie auf Basis bereits vorhandener Strukturinformation zu erstellen. Beispiele für solche strukturierten Daten sind Datenbankschemas, existierende Ontologien und Wissensbasen (z.B. WordNet).

Das Design einer Ontologie ist in der Regel ein iterativer Prozess, selten ist es der Fall, dass eine initiale Ontologie-Version ein perfektes Ergebnis liefern wird. Oft müssen neue Klassen zusätzlich eingebaut werden, verschiedene Eigenschaften anders eingeschränkt oder auch das komplette Konzept überarbeitet werden. Es sollte stets bedacht werden, dass eine Ontologie eine konzeptionelle Modellierung des Weltausschnitts darstellt. Daher sollten Konzepte so gewählt werden, dass sie reale Objekte widerspiegeln, dies erleichtert den Design-Prozess enorm.

*E. Reasoning in Ontologien*

Mit Ontologien können Abhängigkeiten zwischen Ressourcen (Konzepte, Instanzen, etc.) ausgedrückt werden, nicht aber logische Schlüsse oder Handlungsanweisungen. Damit logische Schlussfolgerungen realisiert werden können, werden daher verschiedene Arten von Logiken benötigt

(z.B. Aussagenlogik, Prädikatenlogik, F-Logik, etc.). Werkzeuge die fähig sind, logische Schlussfolgerungen zu bewerkstelligen, werden "Reasoner" genannt. Diese bewegen sich im Themenfeld des automatisierten logischen Schließens von Wissen mit Ontologien. Dazu gehört einerseits die Prüfung der Konsistenz von Wissensmodellen, andererseits das Ableiten von implizitem Wissen, gemäß den Gesetzen der Logik. Das folgende (fiktive) Szenario verdeutlicht die Aufgabe eines Reasoners.

Seien:

- $\mathcal{U}$  der Benutzer, dessen Rechner beschlagnahmt wurde (es gelte die Annahme, der Rechner wurde nicht kompromittiert),
- $\mathcal{I}$  das Image des Rechners von  $\mathcal{U}$  (der Einfachheit halber beinhaltet  $\mathcal{I}$  nur eine Systempartition),
- $\mathcal{B}$  der Beweis der gefunden werden soll (z.B. kinderpornographische Dateien)

sowie die folgenden Prämissen gegeben:

- $\varphi_1 := \mathcal{B}$  wird innerhalb  $\mathcal{I}$  gefunden,
- $\varphi_2 := \mathcal{U}$  ist als alleiniger Administrator erfasst,
- $\varphi_3 := \mathcal{B}$  wurde unter Kennung eines Administrators angelegt
- $\varphi_4 :=$  Logdateien bestätigen den Download von  $\mathcal{B}$  (idealerweise mehrere Downloads aus verschiedenen Quellen)
- $\varphi_5 := \mathcal{B}$  wurde in ausgehenden Datenstreams (z.B. TCP/UDP-Pakete) gefunden

Daraus leitet der Reasoner die folgenden Konklusionen ab:

$$\frac{\psi_3}{\psi_1, \psi_2} \quad \frac{\psi_3}{\psi_1, \psi_2} \\ \hline \varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5$$

und erhält dadurch die impliziten Informationen:

- $\psi_1 := \mathcal{B}$  wurde bewusst von  $\mathcal{U}$  heruntergeladen
- $\psi_2 := \mathcal{B}$  wurde bewusst von  $\mathcal{U}$  weitergegeben
- $\psi_3 := \mathcal{B}$  hat eine Straftat nach §184b und §184d (StGB) begangen (hier muss der Reasoner natürlich erst das explizite Wissen über die Gesetzgebung kennen, um schlussfolgern zu können, welche Art von Straftat hier vorliegt)

Das Ziehen derartiger Schlussfolgerungen ist die zentrale Aufgabe eines Reasoners, zusätzlich jedoch ist dieser auch in der Lage, Beziehungen zwischen einzelnen Entitäten zu erkennen und entsprechend mit verschiedenen Objekten umzugehen. Weiterhin können Daten aus verschiedenen heterogenen Wissensquellen zusammengeführt und Unstimmigkeiten, bzw. Widersprüche darin aufgedeckt

werden. Dies kommt beispielsweise dann zur Geltung, falls die Ontologien von unterschiedlichen Autoren erstellt wurden, die über keinen einheitlichen Wissensstand verfügen.

In der Praxis existieren bereits verschiedene Implementierungen für Reasoner, die die genannten Aufgaben erfolgreich bewältigen können, beispielsweise:

- **FaCT++:** Entwickelt von Dmitry Tsarkov an der University of Manchester (implementiert in C++), basiert auf die etablierten FaCT Reasoning Algorithmen.
- **Racer:** Entwickelt von Volker Haarslev, Concordia University, Canada sowie Ralf Möller, TU Hamburg-Harburg.
- **Pellet:** Entwickelt von Evren Sirin und Bijan Parsia (implementiert in Java), Versteht OWL Lite und OWL DL Syntax.
- **KAON2:** Entwickelt im Zusammenarbeit des Forschungszentrums (FZI), Universität Karlsruhe sowie der University of Manchester, Infrastruktur für OWL-DL, SWRL, F-Logik Ontologien.

Um Rückschlüsse ziehen zu können, muss ein Reasoner fähig sein die Beschreibungssprache zu verstehen, in der eine Ontologie verfasst wurde. Wie bereits erwähnt, ist OWL eine derartige Beschreibungssprache. Von OWL gibt es nach [42] insgesamt drei Versionen mit unterschiedlicher Mächtigkeit. Eine größere Mächtigkeit geht dabei mit einer größeren Komplexität der Berechnungen von Reasonern einher. Benutzer können so eine Sprachversion auswählen, die für Ihre Anforderungen in Sachen Ausdrucksstärke und Berechnungskomplexität am besten geeignet ist. Die drei Sprachversion die zur Auswahl stehen sind:

- **OWL Lite:** dient vor allem zur Erstellung von Taxonomien, also einfachen hierarchisch aufgebauten Ontologien, und stellt nur sehr elementare Beschränkungsöglichkeiten bereit.
- **OWL DL:** ist ausdrucksstärker als OWL Lite und unterstützt dabei deutlich mehr Sprachkonstrukte. Allerdings können diese nur unter Einschränkungen benutzt werden. So darf eine Klasse z.B. keine Instanz einer anderen Klasse sein. Dafür können Reasoner optimal Schlüsse über diese Sprache schließen, da OWL DL gleichzeitig vollständig und entscheidbar ist.
- **OWL Full:** enthält alle Konstrukte der Sprache OWL, ohne Einschränkung bzgl. der Benutzung. So können hier z.B. Klassen sowohl als die Menge aller zugehörigen Individuen, als auch selbst als Individuum betrachtet werden. Während OWL DL und OWL Lite verschiedene Arten von Eigenschaften unterscheiden, die vom Typ des Subjekts und des Objekts abhängen, sind hier alle Arten von Eigenschaften äquivalent. Der entscheidende Nachteil von OWL Full ist jedoch, dass keine Garantien über die Berechnungszeit von Reasonern abgegeben werden können.

F. Semantische Abfragesprachen für Ontologien

So mächtig die Reasoner Werkzeuge auch sind, eine wesentliche Eigenschaft besitzen sie nicht, dass reine Abfragen von Informationen innerhalb der Ontologie. Dafür wiederum sind die Abfragesprachen zuständig, z.B.:

- SPARQL: Eine verbreitete Sprache ist SPARQL. Wie der Name es vermuten lässt, besteht eine Anlehnung der Syntax an die von SQL.
- nRQL: Die "new Racer Query Language", kurz nRQL ist eine eigens für den RacerPro Reasoner entwickelte Abfragesprache, welche insbesondere darauf zielt, sehr komplexe Anfragen an den Reasoner zu stellen, die so mittels anderer Abfragesprachen nicht oder nur auf Umwegen möglich wären.

Folgendes Beispiel demonstriert eine kleine Anfrage in SPARQL, um die Kurzinformationen aller Kriminellen Personen, die am 26.06.1978 geboren wurden, anzeigen zu lassen:

```
PREFIX data: <http://127.0.0.1/BeispielOntologie#>
SELECT ?name ?abstract
WHERE
{
  ?name rdf:type data:criminal ;
  data:abstract ?abstract ;
  data:birth "1978-26-06"^^xmls:date .
}
```

Je nach Ontologie und Granularität der Abfrage, können wertvolle Informationen festgestellt werden, die eigentlich nur mit Einsatz eines "echten" Datenbanksystems möglich gewesen wären.

VIII. ONTOLOGIEN & FORSCHUNG

Prominente Beispiele europäischer Forschungsaktivitäten auf dem Gebiet der Ontologien beinhalten das "On-To-Knowledge" [43] Projekt und das OntoWeb [44] Netzwerk. Im "On-To-Knowledge" Projekt wurden von diversen Firmen fortgeschrittene Ontologie Werkzeuge begleitet von auf Ontologien basierenden Wissensmanagementanwendungen erzeugt. Beide wurden in verschiedenen Fallstudien in der Industrie angewandt und ausgewertet, um das Potential und den zukünftigen Wert dieser Technik zu zeigen.

IX. EXPERTENSYSTEME FÜR DIE SEMANTISCHE ANALYSE

In diesem Abschnitt wird eine abstrakte Sicht auf bereits implementierte Expertensysteme präsentiert. Ziel hierbei ist auf Implementierungsdetails zu verzichten und stattdessen die Kernidee der Systeme hervorzuheben. Hierfür wird Bezug auf alle bisher genannten Methoden und Konzepte genommen, sodass ersichtlich wird, wie diese zusammenhängen.

Der Begriff "Expertensystem" als solches ist nicht einheitlich definiert, daher wird versucht dieses zu umschreiben. Ein typisches Expertensystem versucht, Gedankengänge und Erfahrungen von Experten bestimmter

Fachgebiete auf eine Menge von formalisierten, maschinenverarbeitbaren Operationen abzubilden, um Aspekte einer Problemlösungskompetenz zu reproduzieren und Anwendern zur Verfügung zu stellen. Expertensysteme haben ihren Ursprung in der Disziplin der künstlichen Intelligenz und können wie folgt schematisch skizziert werden:

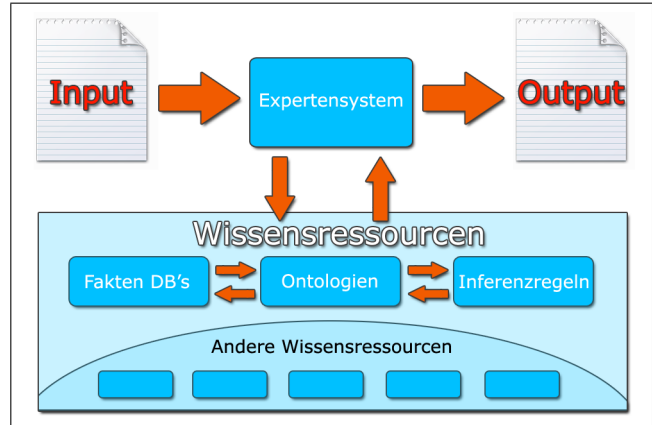


Fig. 12. Eine mögliche Zusammensetzung eines Expertensystems

Die Abbildung kann so verstanden werden, zunächst formuliert ein Benutzer seine Anfrage als Input an das System. Dieses übersetzt die Anfrage in eine maschinelle Repräsentation, sodass durch die verfügbaren Wissensquellen versucht wird, Antworten auf Teile der Frage zu ermitteln. Anschließend werden alle generierten Antworten zusammengeführt und als Output ausgegeben. Die Wissensquellen, auf die das Expertensystem dabei zugreift können sehr unterschiedlich ausfallen. So können z.B. nur Ontologien allein verwendet werden oder jedoch, wie in den meisten Fällen, unterschiedliche Quellen gleichzeitig (beispielsweise Fakten-Datenbanken, Regelsysteme, Wortnetzen, etc.)

Kriminalistische Expertensysteme, so wie diese für die digitale Forensik eingesetzt werden, sind eine Spezialisierung der "herkömmlichen" Expertensysteme. Diese zeichnen sich durch folgende Ziele aus:

- Entdeckung und Klassifizierung von Wissen
- Lernen von Taxonomien
- Charakterisierung
- Generierung

Die Entdeckung und Klassifizierung von Wissen ist der Prozess des Sammelns digitaler Spuren (in Form von Daten) aus unterschiedlichen Quellen (siehe Einleitung). In erster Linie ist hier das Festplatten-Image, des zu untersuchenden Computers von Interesse. Zusätzlich werden Images von andere Medien betrachtet, welche für die forensische Analyse relevante Informationen beinhalten könnten.

Neben Images können auch Rohdaten für die Analyse

hinzugefügt werden. Dabei können die Daten auch in "Chunks", also Fragmente aufgeteilt sein, da es nicht immer Fall ist, dass vollständige Dateien für die Analyse zur Verfügung stehen, insbesondere dann nicht, wenn davor ein Carving-Prozess stattgefunden hat, um gelöschte Dateien wiederherzustellen.

Nachdem die Daten, bzw. Chunks in eine Wissensdatenbank zusammengeführt wurden, werden diese thematisch zu verschiedenen Kategorien zugewiesen, um daraus die vorhandenen (ungeordneten) Datenmengen unterteilen und hierarchisch strukturieren zu können. Die Zuweisung zu den Kategorien erfolgt dabei meistens durch sogenannte Klassifizierungsalgorithmen (z.B. *Naive Bayes*, *Support Vector Machines*, *k-Nearest-Neighbor*, etc.) Diese werden auch "Klassifizierer" genannt und entscheiden anhand statistischer Modelle, zu welchen Kategorien die extrahierten Informationen zugewiesen werden sollen.

Klassifizierer kommen in der Praxis sehr häufig zum Einsatz, um beispielsweise in einem Dokument die verwendete Sprache oder die Thematik darin zu ermitteln. Eines der prominentesten Anwendungen von Klassifizierern ist z.B. die Entscheidung zu treffen, ob eine Mail als "Spam" oder "Nicht-Spam" eingestuft werden soll. Ein wichtiger Klassifizierer der hier erwähnt werden soll, ist der sogenannte *Naive Bayes*. Dieser hat zwei wesentliche Vorteile, die seine Anwendung erst so interessant machen:

- 1) Obwohl Naive Bayes Annahmen trifft, die in der Realität so gut wie nie wahr sind ("Alle Attribute / Wörter sind stochastisch voneinander unabhängig"), erzielt dieser in vielen Fällen gute Erkennungsraten. Ein Grund hierfür ist, dass keine exakten Wahrscheinlichkeitsschätzungen benötigt werden, solange die maximale Wahrscheinlichkeit der korrekten Kategorie zugewiesen wird.
- 2) Naive Bayes ist bzgl. der Komplexität schnell und einfach zu implementieren und dient in der Praxis für selbstentwickelte Klassifizierer als eine optimale Baseline.

Die Kategorien können neben, rein textuellen Chunks auch Multimedia-Daten beinhalten, die zur selben Thematik der Chunks passen, z.B. Audio- oder Videoaufnahmen über ein Ereignis, zu dem in dem Chunk Bezug genommen wird. Falls mehrere Kategorien entstehen, die in gewisserweise eine Ähnlichkeit zueinander aufweisen, so können diese auch zusammengefasst werden. Dies kann genau dann passieren, falls unterschiedlich kategorisierte Chunks auf das selbe Thema referenzieren. Lässt sich jedoch ein Chunk in mehrere Kategorien gleichzeitig einordnen, sind weitere Klassifizierungsmetriken notwendig, die eine Entscheidung treffen, welche Kategorie am meisten domiert.

Nachdem das Wissen entdeckt und klassifiziert wurde, müssen im nächsten Schritt Taxonomien erlernt werden. Taxonomien definieren Klassen von Objekten und Beziehungen zwischen diesen, enthalten jedoch keine Beschreibungen bzw.

Definitionen von Objekten. Die häufigste Erscheinungsform sind hierarchisch geordnete Begriffsmengen, wobei die Begriffe baumartig angeordnet sind, sodass mit zunehmender Tiefe ihre Spezifität zunimmt. Ein bekanntes Beispiel hierfür sind Betriebssysteme, deren Verzeichnisstruktur eine Taxonomie ausmachen:

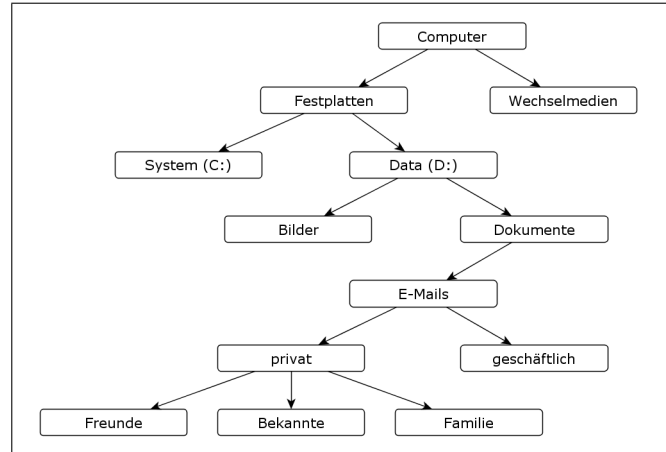


Fig. 13. Eine Verzeichnisstruktur als Taxonomie

Die Abbildung zeigt die fehlende Beschreibungsmöglichkeit gegenüber Ontologien (es gibt beispielsweise als Eigenschaften nur die *ist\_Ein* Beziehung), wobei Ontologien praktisch jede Art von Beziehung zulassen, da sich diese selbst definieren lassen. Für gewöhnlich werden gelernte Taxonomien (je nach Szenario) in Ontologien überführt, um dadurch Reasoning und automatische Abfragen überhaupt erst zu ermöglichen. Die Überführung der Taxonomien zu Ontologien ist eines der Aufgaben der semi-, bzw. vollautomatischen Generierung von Ontologien (siehe dazu [34]).

Nachdem die Ontologien generiert wurden, kann mithilfe von Reasoning begonnen werden, diese auf Unstimmigkeiten zu überprüfen. Wurden keine Widersprüche gefunden, so lässt sich ein Inferenzprozess durchführen, um neues Wissen, bzw. neue Informationen aus den bestehenden abzuleiten. Das "neu"-gewonnene Wissen wird in einer weiteren Wissensdatenbank gespeichert und kann anschließend abgefragt werden. Eine weitere Möglichkeit besteht darin, bestehendes und neues Wissen zu vereinen und diese dann als ein Report ausgeben zu lassen, der alle wichtige Informationen zusammenfasst. Dies ist die Idealvorstellung eines kriminalistischen Expertensystems.

Derartige Systeme wurden der Recherche nach ([32], [36], [37] und [38]) nur innerhalb von Forschungsarbeiten realisiert. Es wird vermutet, dass (noch) keine fertige Lösungen existieren, die in der Praxis mit den genannten Eigenschaften, eingesetzt werden können.

## X. EINFÜHRUNG IN FORENSISCHER LINGUISTIK

Mit Ontologien und Reasoning können Wissensbestände modelliert und erfragt werden. Wenn es jedoch um die

Zuordnung von Daten zu deren Autoren geht, müssen andere Strategien gefunden werden. Eine dieser Strategien stellt die forensische Linguistik dar und soll an dieser Stelle mit den zuvor behandelten Themen in Verbindung gebracht werden.

Die forensische Linguistik ist ein angewandter Bereich der Linguistik im Schnittfeld zwischen "Sprache und Verbrechen" und kann der Recherche nach [40], bis ins Jahr 1887 zurückverfolgt werden, als zum ersten Mal die Zuordnung von Text zu Autoren dokumentiert wurde. Die Disziplin beschäftigt sich insbesondere mit der Thematik der Autorzuordnung von juristisch relevanten Texten, seien es z.B.:

- Erpresserbriefe
- Bekenner schreiben (Terror Anschläge)
- Geständnisse
- Selbstmord schreiben (Authentizität)
- Zeugenaussagen
- Testamente
- Plagiate
- etc.

Die Disziplin behilft sich mit stilometrischen Methoden, um zu gegebenen Dokumenten entscheiden zu können, ob ein Verdächtiger der tatsächliche Autor des Dokuments ist. Als Metriken für diese Methoden zählen unter anderem: Satzlänge, Häufigkeit von Funktionswörtern, Wortschatz eines Autors, Häufigkeit von Nebensätzen, Länge einzelner Phrase, etc.

Im Rahmen einer forensischen Analyse kann eine mögliche Lösung für das folgende Szenario, mithilfe der forensischen Linguistik gefunden werden:

Ermittler finden bei einer Hausdurchsuchung ein Handy von dem Sie wissen, dass dieser einer Person  $\mathcal{X}$  geklaut worden ist. Von dem Gerät aus wurden mehrere SMS Nachrichten mit Morddrohungen (oder ähnliches) an eine Zielperson verschickt. Die Annahme der Ermittler ist, dass der Täter  $\mathcal{Y}$ , bei dem das Gerät gefunden wurde, diese Nachrichten verschickt hat. Dies jedoch können sie nicht beweisen, da weder der Zeitpunkt des Diebstahls bekannt ist noch eine genaue Lokalisierung des Ortes von wo die Nachricht abgeschickt wurde (für das Szenario wird angenommen  $\mathcal{X}$  und  $\mathcal{Y}$  teilen sich die selbe Cell-ID). Den Ermittlern bleibt damit nur noch die Möglichkeit, anhand der verfassten Nachrichten, urteilen zu können, ob  $\mathcal{X}$  oder  $\mathcal{Y}$  diese verschickt hat. Als mögliche Lösung für diese Problematik bieten sich die stilometrischen Methoden der forensischen Linguistik an, welche darauf zielen Muster im Stil der Nachrichten zu bestimmen. Diese Muster können dafür benutzt werden, um erkennen zu können, ob  $\mathcal{X}$  oder  $\mathcal{Y}$  der jeweilige Autor ist.

Dies stellt natürlich nur ein fiktives Beispiel für die Anwendung der forensischen Linguistik dar, es gibt viele weitere (reale) Beispiele, welche jedoch aufgrund der Aktualität dieser Disziplin nicht ausreichend recherchiert werden konnten.

## XI. FAZIT - CARVING

Heutige Carving Tools verwenden verschiedene Carving Techniken. Allerdings ist bei den Carving Tools wie EnCase Forensic oder Forensik Toolkit, die keine Open Source Tools sind, nicht klar, welche Algorithmen genau benutzt werden. Es gibt viele unterschiedliche Carving Methoden und die Carving Tools haben unterschiedliche Features. Dies kann bewirken, dass sich die Carving Ergebnisse der Tools unterscheiden. Für eine größtmögliche Datenwiederherstellung könnte in Erwägung gezogen werden, mehrere Tools auf verschiedenen Systemen parallel auszuführen.

Carving Tools leisten schon viel, müssen aber stetig weiterentwickelt werden. Da die Datenträger eine immer größere Kapazität bieten, werden Carving Algorithmen benötigt, die skalieren und dennoch präzise sind. Wir haben gesehen, dass die Wiederherstellung von fragmentierten Dateien schwierig ist. Obwohl i.d.R. nur ein relativ kleiner Teil der Dateien auf Datenträgern fragmentiert ist, ist es wichtig diese wiederherzustellen, da die für forensische Zwecke besonders interessanten Dateiformate mit größerer Wahrscheinlichkeit fragmentiert sind als die weniger relevanten Dateiformate. Weiterhin ist es möglich, dass zukünftig mehr Datenträger mit starker Fragmentierung existieren werden, da vermutlich zunehmend herkömmliche Festplatten durch SSDs ersetzt werden, die eine höhere Geschwindigkeit bieten und zudem immer günstiger zu erwerben sind.

Die Hersteller der bekannten Carving Tools Encase, Forensic Toolkit und X-Ways Forensics achten schon darauf dass die Ergebnisse ihrer Tools vor Gericht verwertbar sind. Die Vorgaben vom Gesetzgeber geben allerdings keine konkrete Handlungsanweisung. Beispielsweise soll die Erhebung kernbereichsrelevanter Daten, soweit informationstechnisch und ermittlungstechnisch möglich, unterbleiben [6]. Welche Daten das aber genau sind, bzw. wo der absolut geschützten Kernbereich privater Lebensgestaltung anfängt, wird nicht konkret erläutert.

## XII. FAZIT - SEMANTISCHE ANALYSE

In der vorliegenden Ausarbeitung wurden Methoden und Konzepte der semantischen Analyse in der digitalen Forensik untersucht. Es wurde erläutert, welche Möglichkeiten verwendet werden können, um relevante Informationen aus großen Datenmengen zu extrahieren, sodass diese z.B. mithilfe von Ontologien wiederverwendet werden können. Auf die Thematik der Ontologien wurde anschließend ebenfalls eingegangen, wobei hier die wesentlichen Punkte fokussiert wurden, die im Kontext der forensischen Analyse eine Rolle spielen. Unter anderem wurde die Idee des Reasonings anhand eines fiktiven Szenarios beschrieben, welches es Ermittlern erlaubt aus gegebenen Fakten, die zuvor aus Dateien extrahiert und mittels einer formalen Beschreibungssprache kodiert wurden, implizites Wissen abzuleiten. Außerdem wurden thematisch verwandte Gebiete der semantischen Analyse erwähnt, die es Ermittlern ermöglichen sollen, die Zuordnung von Autoren zu bereits gefundenen digitalen Spuren zuzuordnen.

Insgesamt lässt sich festhalten, dass die semantische Analyse

ein ausgesprochen spannendes und hochaktuelles Forschungsfeld innerhalb der digitalen Forensik darstellt, welches zahlreiche Disziplinen kombiniert, die teilweise aktuell selbst noch erforscht werden (z.B. Ontology Engineering, Text Mining, künstliche Intelligenz, etc.) Zu erwähnen wäre noch die Tatsache, dass es nicht faktisch belegt werden konnte, ob tatsächlich praxistaugliche Lösungen existieren, um den gesamten semantischen Analyseprozess zu automatisieren. Daher besteht die Annahme, dass eine Interaktion zwischen Ermittler und Maschine bestehen muss, um relevante digital Beweise auffindig zu machen. Weiterhin erfordert der technische Aspekt der semantische Analyse die Zusammenarbeit unterschiedlicher Kompetenzen (Forensiker, Informatiker, Computeringuisten, etc.), den nur damit können zukünftige Systeme weiterentwickelt werden, die ein gemeinsames Expertenwissen voraussetzen.

#### DANKSAGUNG

Ein ganz besonderer Dank gilt unserer Betreuerin, Fr. Margarida de Castro Neves (Fraunhofer IGD, TU-Darmstadt) die uns durch ihre hilfreichen Anregungen unterstützt hat und großes Interesse an dieser Ausarbeitung zeigte.

#### XIII. LITERATURVERZEICHNIS

##### REFERENCES

- [1] A. J. Marcella und D. Menendez, *Cyber forensics: a field manual for collecting, examining, and preserving*. : Boca Raton, USA: Auerbach Publications, 2008. Online verfügbar unter: <http://books.google.de/books?id=enEqHuVht7HgC&printsec=frontcover&dq=Cyber+forensics:+a+field+manual+for+collecting,+examining,+and+preserving>, zuletzt abgerufen am 10.07.2010. S. 73. 2
- [2] A. Pal and N. W. Memon, *The Evolution of File Carving*. New York, USA: Signal Processing Magazine, IEEE, 2009. Online verfügbar unter: <http://digital-assembly.com/technology/research/pubs/ieeee-spm-2009.pdf>, zuletzt abgerufen am 10.07.2010. 1, 2, 3, 5, 6, 7, 8, 9
- [3] A. Pal and N. Memon, *Automated reassembly of file fragmented images using greedy algorithms*. : New York, USA: IEEE, 2006. Online verfügbar unter: [http://www.znu.ac.ir/data/members/fazli\\_saeid/DIP/Paper/ISSUE2/01576811.pdf](http://www.znu.ac.ir/data/members/fazli_saeid/DIP/Paper/ISSUE2/01576811.pdf), zuletzt abgerufen am 10.07.2010. 7
- [4] A. Pal, K. Shanmugasundaram und N. Memon, *Automated Reassembly of Fragmented Images*. : New York, USA: Polytechnic University, 2003. Online verfügbar unter: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.10.8473&rep=rep1&type=pdf>, zuletzt abgerufen am 10.07.2010. 6, 7, 8
- [5] B. Coppin, *Artificial intelligence illuminated*. : Sudbury, USA: Jones & Bartlett Learning, 2004. Online verfügbar unter: [http://books.google.de/books?id=LcOLqodW28EC&pg=PA153&dq=Alpha-Beta-Pruning&hl=de&ei=GmA-TLLHhN3NjAeJ5Pj4Aw&sa=X&oi=book\\_result&ct=result&resnum=3&ved=0CDUQ6AEwAg#v=onepage&q=Alpha-Beta-Pruning&f=false](http://books.google.de/books?id=LcOLqodW28EC&pg=PA153&dq=Alpha-Beta-Pruning&hl=de&ei=GmA-TLLHhN3NjAeJ5Pj4Aw&sa=X&oi=book_result&ct=result&resnum=3&ved=0CDUQ6AEwAg#v=onepage&q=Alpha-Beta-Pruning&f=false), zuletzt abgerufen am 10.07.2010. S. 153. 7
- [6] Bundesverfassungsgericht, *Vorschriften im Verfassungsschutzgesetz NRW zur Online-Durchsuchung und zur Aufklärung des Internet nichtig*. : Karlsruhe, Deutschland: Bundesverfassungsgericht, 2008. Online verfügbar unter: <http://www.bundesverfassungsgericht.de/pressemitteilungen/bvg08-022>, zuletzt abgerufen am 10.07.2010. 4, 19
- [7] Bundesverfassungsgericht, *Anforderungen an die Beschlagnahme von Datenträgern und hierauf gespeicherter Daten*. : Karlsruhe, Deutschland: Bundesverfassungsgericht, 2005. Online verfügbar unter: [http://www.bundesverfassungsgericht.de/bverfg\\_cgi/pressemitteilungen/frames/bvg05-047](http://www.bundesverfassungsgericht.de/bverfg_cgi/pressemitteilungen/frames/bvg05-047), zuletzt abgerufen am 10.07.2010. 4
- [8] DigitalAssembly, *Adroit Photo Forensics 2010*. : New York, USA: DigitalAssembly, 2010. Online verfügbar unter: <http://digital-assembly.com/support/adroit-photo-forensics/apf-user-manual.pdf>, zuletzt abgerufen am 10.07.2010. 10
- [9] E. W. Dijkstra, *A note on two problems in connexion with graphs*. : Amsterdam, Niederlande: Mathematisch Centrum, 1959. Online verfügbar unter: <http://www-m3.ma.tum.de/foswiki/pub/MN0506/WebHome/dijkstra.pdf>, zuletzt abgerufen am 10.07.2010. 7
- [10] Forensics Wiki, *A note on two problems in connexion with graphs*. Online verfügbar unter: [http://www.forensicswiki.org/wiki/File\\_Carving](http://www.forensicswiki.org/wiki/File_Carving), zuletzt abgerufen am 10.07.2010. 4
- [11] Forensic Toolkit, *THE INDUSTRY-STANDARD COMPUTER FORENSICS SOFTWARE USED BY GOVERNMENT AGENCIES AND LAW ENFORCEMENT AROUND THE WORLD*. : Lindon, USA: AccessData, 2010. Online verfügbar unter: <http://www.accessdata.com/forensictoolkit.html>, zuletzt abgerufen am 10.07.2010. 4
- [12] Forensic Toolkit, *Distributed Processing... Going beyond commonly accepted boundaries*. : Lindon, USA: AccessData, 2010. Online verfügbar unter: [http://www.accessdata.com/distributed\\_processing.html](http://www.accessdata.com/distributed_processing.html), zuletzt abgerufen am 10.07.2010. 3
- [13] G. G. Richard III, V. Roussev, und L. Marziale, *In-Place File Carving*. : New Orleans, USA: University of New Orleans, 2007. Online verfügbar unter: <http://cs.uno.edu/~golden/Stuff/ifip2007-final.pdf>, zuletzt abgerufen am 10.07.2010. 9
- [14] Guidance Software Inc., *EnCase Forensic Verion 6*. New Berlin, USA: Digital Intelligence, 2009. Online verfügbar unter: <http://www.digitalintelligence.com/software/guidancesoftware/encase/>, zuletzt abgerufen am 10.07.2010. 2
- [15] Guidance Software Inc., *EnCase® Forensic for Law Enforcement*. : Amsterdam, Niederlande: Guidance Software, 2009. Online verfügbar unter: <http://www.guidancesoftware.com/WorkArea/DownloadAsset.aspx?id=674>, zuletzt abgerufen am 10.07.2010. 4
- [16] H. Allen, *Introduction to FreeBSD, PacNOG I Workshop, Additional Topics*. Nadi, Fiji: Network Startup Resource Center, 2005. Online verfügbar unter: <http://www.pacnog.org/pacnog1/day1/freebsd/intro-freebsd-additional-topics.pdf>, zuletzt abgerufen am 10.07.2010. 3
- [17] J. G. Cleary und W. J. Teahan, *Unbounded length context for ppm*. : Hamilton, Neu Seeland: University of Waikato, 1997. Online verfügbar unter: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.402&rep=rep1&type=pdf>, zuletzt abgerufen am 10.07.2010. 6
- [18] J. M. Claar et al, *File system block sub-allocator*. : USA: 2000. Online verfügbar unter: <http://www.google.com/patents?vid=6041407>, zuletzt abgerufen am 10.07.2010. 3
- [19] J. Metz, *Shrinking the gap: carving NTFS-compressed files*. : Forensic Focus, 1959. Online verfügbar unter: <http://www.forensicfocus.com/carving-ntfs-compressed-files>, zuletzt abgerufen am 10.07.2010. 9
- [20] K. M. J. Lofgren et al, *Wear leveling techniques for flash EEPROM systems*. : USA: 2005. Online verfügbar unter: <http://www.google.com/patents?vid=6850443>, zuletzt abgerufen am 10.07.2010. 3
- [21] L. Marziale, G. G. Richard III, V. Roussev, *Massive threading: Using GPUs to increase the performance of digital forensics tools*. New Orleans, USA: Elsevier, 2007. Online verfügbar unter: <http://www.dfrws.org/2007/proceedings/p73-marziale.pdf>, zuletzt abgerufen am 10.07.2010. 3
- [22] M. Zelkowitz, *Advances in Computers: Software Development*. : College Park, USA: Academic Press, 2008. Online verfügbar unter: <http://books.google.de/books?id=zIYH5Z8mVYC&printsec=frontcover&dq=Advances+in+Computers+Software+Development+Von+Marvin+Zelkowitz>, zuletzt abgerufen am 10.07.2010. S. 7. 2
- [23] O. Diedrich, *Tuning the Linux file system Ext3*. : The H Open Source, 2008. Online verfügbar unter: <http://www.h-online.com/open/features/Tuning-the-Linux-file-system-Ext3-746480.html>, zuletzt abgerufen am 10.07.2010. 1
- [24] P. Baumann und N. Rahn, *Digitale Forensik: Einführung in die Thematik*. : Darmstadt, Deutschland: Fraunhofer IGD, 2010. Online verfügbar unter: [http://www.igd.fraunhofer.de/~pebinger/lectures/digitalforensics/sose2010/slides/01\\_Einfuehrung.pdf](http://www.igd.fraunhofer.de/~pebinger/lectures/digitalforensics/sose2010/slides/01_Einfuehrung.pdf), zuletzt abgerufen am 10.07.2010. 4
- [25] P. Ebinger, *Carving und semantische Analyse in der digitalen Forensik*. Darmstadt, Deutschland: Fraunhofer-Institut für Graphische Datenverarbeitung, 2010. Online verfügbar unter: [http://www.igd.fraunhofer.de/~pebinger/lectures/digitalforensics/sose2010/digital\\_forensics\\_syllabus/node17.html](http://www.igd.fraunhofer.de/~pebinger/lectures/digitalforensics/sose2010/digital_forensics_syllabus/node17.html), zuletzt abgerufen am 10.07.2010. 1
- [26] P. Neugebauer und T. Volk, *Hostforensik*. : Darmstadt, Deutschland: Fraunhofer IGD, 2010. Online verfügbar unter: [http://www.igd.fraunhofer.de/~pebinger/lectures/digitalforensics/sose2010/slides/02\\_Host-Forensik.pdf](http://www.igd.fraunhofer.de/~pebinger/lectures/digitalforensics/sose2010/slides/02_Host-Forensik.pdf), zuletzt abgerufen am 10.07.2010. 4
- [27] S.J.J. Kloet, *Measuring and Improving the Quality of File Carving Methods*. Almere, Niederlande: Eindhoven University of Technology,

2007. Online verfügbar unter: [http://www.forensicswiki.org/w/images/b/b9/Kloet\\_2007.pdf](http://www.forensicswiki.org/w/images/b/b9/Kloet_2007.pdf), zuletzt abgerufen am 10.07.2010. S. 4, 79. 3, 4
- [28] S. L. Garfinkel, *Carving contiguous and fragmented files with fast object validation*. Monterey, USA: Elsevier, 2009. Online verfügbar unter: <http://www.dfrws.org/2007/proceedings/p2-garfinkel.pdf>, zuletzt abgerufen am 10.07.2010. 2, 3, 4, 5, 6, 9
- [29] Western Digital, *SSD SolidStor Technology Overview*. : Lake Forest, USA: Western Digital, 2001-2010. Online verfügbar unter: <http://www.wdc.com/en/products/ssd/technology.asp?id=3>, zuletzt abgerufen am 10.07.2010. 3
- [30] Wikipedia, *Disk structure (Abbildung)*. : Wikipedia, 2010. Online verfügbar unter: [http://en.wikipedia.org/wiki/Data\\_cluster](http://en.wikipedia.org/wiki/Data_cluster), zuletzt abgerufen am 10.07.2010. 2
- [31] X-Ways Software Technology AG, *X-Ways Forensics: Integrierte Software für Computerforensik*. : Köln, Deutschland: X-Ways, 1959. Online verfügbar unter: <http://www.x-ways.net/forensics/index-d.html>, zuletzt abgerufen am 10.07.2010. 4
- [32] V. Loia, M. Mattiucci, S. Senatore, M. Veniero, *Computer Crime Investigation by means of Fuzzy Semantic Maps* (Kapitel: "Ontology"), IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies - Workshop, 2009. 12, 18
- [33] T. Gruber, *Encyclopedia of Database Systems* (Kapitel: "Ontology"), Ling Liu and M. Tamer Özsu (Eds.), Springer-Verlag, 2009. 14
- [34] M. Ahrens, *Semi-automatische Generierung einer OWL-Ontologie aus domänenspezifischen Texten am Beispiel von HUMINT-Meldungen*, Masterarbeit, Universität Bonn, 2009. 15, 18
- [35] R. Smith, *An approach to protecting against unknown vulnerabilities*, HP Labs, Systems Security Lab, 2008. 11
- [36] B. Schatz, G. Mohay, A. Clark, *Rich Event Representation for Computer Forensics*, Proceedings of the 2004 Asia Pacific Industrial Engineering and Management Systems (APIEMS 2004), Gold Coast, Australia, 2004. 18
- [37] A. M. Hoss, D. L. Carver, *Weaving ontologies to support digital forensic analysis*, Proceedings of the 2009 IEEE international conference on Intelligence and security informatics, IEEE Press Piscataway, NJ, USA, 2009. 11, 18
- [38] E. Casey, *Digital Evidence and Computer Crime*, cmdLabs, Baltimore, MD, USA, 2004. 11, 18
- [39] J. Huang, A. Yasinsac, P. J. Hayes *Knowledge Sharing and Reuse in Digital Forensics*, sadfe, pp.73-78, 2010 Fifth IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering, 2010. 11
- [40] DIMACS/IIT Authorship Attribution Working Group, *Identifying real-life authors in massive document collections*, DIMACS Center, Rutgers University, Piscataway, NJ, 2006. 19
- [41] Digital Forensic Research Workshop (DFRWS), *About: "DFRWS"*, <http://www.dfrws.org>, zugegriffen am: 30.06.2010. 11
- [42] M.K. Smith, C. Welty, D.L. Mc Guinness (2004): *OWL Web Ontology Language Guide*, W3C Recommendation, <http://www.w3.org/TR/2004/REC-owl-guide-20040210>, zugegriffen am: 30.06.2010. 16
- [43] On-To-Knowledge, *Content-driven Knowledge-Management through Evolving Ontologies*, EU-IST Project IST-1999-10132. 17
- [44] Institut für Angewandte Informatik und Formale Beschreibungsverfahren (AIFB): *OntoWeb*, <http://www.aifb.kit.edu/web/OntoWeb>, zugegriffen am: 30.06.2010. 17
- [45] The World Wide Web Consortium (W3C), *Standards: "Semantic Web"*, <http://www.w3.org/standards/semanticweb>, zugegriffen am: 30.06.2010. 14
- [46] Extensible Markup Language for Discourse Annotation (EXMARaLDA), *Hilfe: "Dokumentation"*, <http://www.exmaralda.org>, zugegriffen am: 30.06.2010. 14
- [47] Transcribe & Analyze (Transana), *About: "Transana"*, <http://www.transana.org>, zugegriffen am: 30.06.2010. 14
- [48] H. Hippner, R. Rentzmann, *Text Mining*, Erschienen in: Informatik Spektrum, Volume 29, Nr. 4, Sprinter Verlag, Heidelberg, Deutschland, Seite 287-290, im Jahr: 2006. 12
- [49] M. A. Hearst, *Untangling Text Data Mining*, Proceedings of ACL'99: the 37th Annual Meeting of the Association for Computational Linguistics, University of Maryland, June 20-26, 1999. 12
- [50] J. Han, M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann Series in Data Management Systems, 2006. 12