



Erste Bachelorarbeit

Graphical Harddisk Forensics

Ausgeführt am Fachhochschul-Bachelorstudiengang „IT-Security“
an der Fachhochschule St. Pölten

unter der Leitung von

Prof. (FH) Dipl.-Ing. Bernhard Fischer

ausgeführt von

Christoph Pritz
IS081019

St. Pölten, am 22. Juni 2010

Ehrenwörtliche Erklärung

Ich versichere, dass

- ich diese Bachelorarbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich sonst keiner unerlaubten Hilfe bedient habe.
- ich dieses Bachelorarbeitsthema bisher weder im Inland noch im Ausland einem Begutachter/einer Begutachterin zur Beurteilung oder in irgendeiner Form als Prüfungsarbeit vorgelegt habe.
- diese Arbeit mit der vom Begutachter/von der Begutachterin beurteilten Arbeit übereinstimmt.
- Ich räume hiermit der Fachhochschule St. Pölten das ausschließliche und räumlich unbeschränkte Werknutzungsrecht für alle Nutzungsarten an dieser Bachelorarbeit ein, und behalte das Recht, als Urheber dieses Werkes genannt zu werden.

Altwaldhäusl, am 22. Juni 2010

Ort, Datum

Unterschrift

Abstract

In der heutigen Welt der Information ist ein Datenträger für Jedermann eine Selbstverständlichkeit. Unzähligen Festplatten, USB Sticks, CDs, DVDs begegnen wir bewusst oder unbewusst in unserem normalen Alltag. Mit der großen Menge an Daten stellt man sich jedoch auch die Frage wie man in bestimmten Situationen wie z.B. kriminaltechnischen Fälle oder Rettungsmaßnahmen von beschädigten Datenträgern den Überblick behalten kann. Aufbauend auf diese Frage wurde das Projekt "Graphical Harddisk Forensics" ins Leben gerufen. Dabei handelt es sich um ein Programm, das den gewünschten Datenträger abhängig von den darauf gespeicherten Daten grafisch darstellt. Die besondere Herausforderung dabei ist die richtige Klassifizierung der Daten anhand bestimmter Merkmale, was auch als Data Mining bezeichnet wird. In dieser Arbeit finden sich verschiedene Algorithmen, die bestimmte Dateitypen (z.B.: ausführbare Dateien, Textdateien, etc...) klassifizieren. Auch auf eine mögliche Implementierung wurde bereits mit ein paar Gedankengängen (z.B. Suchalgorithmus) vorgegriffen.

Inhaltsverzeichnis

1	Einleitung	5
2	Ziele	5
2.1	Forschungsfrage:	6
3	Data Mining	6
3.1	Theoretische Grundlagen	6
3.2	Vorgehensweise für Data Mining Prozeduren	7
3.2.1	Clustering	7
3.2.2	Bayes Verfahren	7
3.2.3	Induktives Lernen	8
3.3	Verteilung der Dateitypen in gebräuchlichen Betriebssystemen	9
3.3.1	Ergebnisse	11
3.4	Data Mining Ansätze für verschiedene Dateitypen	12
3.4.1	Multimediateilnehmend - Bilddateien	12
3.4.2	Ausführbare Dateien	16
3.4.3	Texterkennung	18
4	Implementierung	20
5	Schlussfolgerung und Ausblick	23
6	Quellenverzeichnis	24
7	Abbildungsverzeichnis	25

1 Einleitung

Der Einsatz von Datenträgern ist in der heutigen Zeit aufgrund der großen Menge von Information sehr verbreitet. Viele Menschen besitzt die entsprechende Hardware um Daten persistet zu speichern. Was passiert aber, wenn dieser Datenträger mit wichtigen Daten unbrauchbar bzw. mit normalen Mitteln nicht mehr zugreifbar ist?

Wie kann man dieses Problem nun vereinfachen oder gar lösen? Die Antwort liegt in der "Graphical Harddisk Forensics" bzw. in der grafischen Darstellung von Festplatten, oder allgemein gesagt Datenträgern. Sempel gesagt ist dies ein Programm, dass es einem ermöglicht den Datenträger grafisch darzustellen, wobei die Darstellung von den darauf gespeicherten Daten abhängig ist. Ein wichtiger Aspekt des Programms ist somit die Datenklassifizierung bzw. Mustererkennung, die auch als "Data Mining" bezeichnet wird. Diese basiert meistens auf der Erkennung von individuellen Merkmalen bestimmter Dateitypen wie zum Beispiel Header. Bei anderen Dateien ist die Erkennung meist nur durch statistische Methoden möglich. Durch dieses Programm soll es möglich sein einen Überblick über die auf einem Datenträger gespeicherten Daten zu erlangen, um danach, falls nötig, die Suche auf bestimmte Bereiche einzuschränken. Dies würde viele möglicherweise unerfolgreiche Suchvorgänge von Anfang an ausschließen. Es ist ebenfalls zu erwähnen, dass ein Irrtum bei der Dateiklassifizierung nie zu 100 Prozent ausgeschlossen werden kann. Dies bedeutet, dass Sektoren, bei denen die Bestimmung unklar war, nochmals von Menschenhand überprüft werden sollten um "False Positives" auszuschließen.

2 Ziele

Ziel dieser Arbeit ist es die Basis bzw. die Algorithmen für ein Programm zu finden, welches, wie in der Einleitung erläutert, den Datenträger anhand der darauf gespeicherten Daten grafisch darstellt. Dazu werden die individuellen Eigenschaften (Header, statistische Eigenheiten, etc. . .) recherchiert,

die benötigt werden um die Daten zu klassifizieren. Es ist vorgesehen die Festplatte dabei nicht Datei für Datei, sondern aufgrund der Einfachheit Block für Block abzutasten. Die Blockgröße ist vom verwendeten Dateisystem (FAT32, NTFS, ReiserFS,...) abhängig. Diese Arbeit soll und kann in keinem Fall die Erkennung jeder mögliche Dateitypen abdecken, es soll jedoch ein Grundkonzept und ein Anstoß für die Weiterentwicklung dieses Ansatzes gegeben werden.

2.1 Forschungsfrage:

Gibt es Algorithmen zur Klassifizierung von digitalen Rohdaten?

3 Data Mining

3.1 Theoretische Grundlagen

Der Begriff "Data Mining" ist in der Welt der Informationsgesellschaft ein von vielen Leuten benutztes Wort. Es beschreibt "[...] die Extraktion implizit vorhandenen, nicht trivialen und nützlichen Wissens aus großen, dynamischen, relativ komplex strukturierten Datenbeständen" (Bissantz 2010, S. 139). Dabei können die Rohdaten aus jedem beliebigen Bereich unseres Lebens stammen. Data Mining findet dabei Anwendung im Marketing, der Paneldatenforschung, der Kostenrechnung oder dem Helpdesk (vgl. Bissantz 2010, S. 139-140). Für die Anwendung von Data Mining gibt es sicher noch sehr viele andere Anwendungsfälle, in dieser Arbeit konzentriert man sich jedoch auf das Data Mining im Sinne der Computer Forensik. Diese analysiert, wie in der Kriminaltechnik, den Tatort, also den Computer, nach der eigentlichen Tat, um Beweise für die Tat zu sammeln. Hier kann Data Mining eingesetzt werden, um den Vorgang erheblich zu beschleunigen. Hierbei ist es nicht zwingend erforderlich, dass ein Verbrechen im Sinne der Computerkriminalität begangen wurde, sondern Data Mining kann auch bei der Wiederherstellung von verlorenen Daten benutzt werden.

3.2 Vorgehensweise für Data Mining Prozeduren

In diesem Unterkapitel sollen die meist genutzten Verfahren genannt werden, die benutzt werden um die Daten zu klassifizieren.

3.2.1 Clustering

Beim Clustering (vgl. Bissantz 2010, S. 140) wird versucht jedes Element eindeutig einem Cluster zuzuordnen. Einen Cluster kann man sich als eine Gruppe oder Container von gleichen Objekten vorstellen. Das Clustern soll dabei so erfolgen, dass die Ähnlichkeit von Objekten in der gleichen Gruppe möglichst hoch, und die Ähnlichkeit von Objekten in verschiedenen Gruppen möglichst niedrig ist.

Konzeptionelles Clustern

Der Ansatz des konzeptionellen Clustering (vgl. Bissantz 2010, S. 140-141) erweitert das normale Clustern um eine Beschreibung der Cluster. Diese Beschreibung wird als Konzept bezeichnet. Die jeweiligen Cluster besitzen beim konzeptionellen Clustern also eine Beschreibung der darin enthaltenen Objekte. Dies ist in den meisten Fällen praktischer, denn eine reine Ansammlung der Objekte in namenlosen Clustern ist nicht sehr sinnvoll.

3.2.2 Bayes Verfahren

Das Bayes Verfahren (vgl. Bissantz 2010, S. 141-142) wird benutzt, um die einzelnen Objekte mit verschiedenen hohen Wahrscheinlichkeiten den Gruppen zuzuteilen. Die Gruppe mit der höchsten Wahrscheinlichkeit wird der Container des Objekts. Die Wahrscheinlichkeitsberechnung erfolgt durch das Bay'sche Theorem, dass die Berechnung von bedingten Wahrscheinlichkeiten ermöglicht.

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

Somit wird also ein Objekt nicht hundertprozentig einer Gruppe wie beim Clustering zugeteilt, sondern die Zuordnung erfolgt mit einer bestimmten Unsicherheit.

3.2.3 Induktives Lernen

Beim induktivem Lernen (vgl. Bissantz 2010, S. 142) wird versucht, anhand von Objekten, die schon klassifiziert wurden, Regeln abzuleiten, die danach für die Klassifizierung von neuen Objekten verwendet werden können.

Man könnte hier in unserem Beispiel dem Programm eine große Menge an bereits klassifizierte Daten (z.B.: Bilder) vorgeben, die das Programm dann nutzen würde um die individuellen Merkmale der einzelnen Dateitypen zu extrahieren. Das Problem besteht hier nur in dem Algorithmus, der diese Merkmale extrahiert, denn dieser muss dynamisch genug sein um auf individuelle Dateitypen richtig zu reagieren.

3.3 Verteilung der Dateitypen in gebräuchlichen Betriebssystemen

Bevor die eigentliche Untersuchung der Dateitypen stattfindet, muss man sich fragen welche Dateitypen man überhaupt untersuchen will. Dazu wurde eine Analyse durchgeführt welche Dateitypen in gängigen Betriebssystemen in welcher Verteilung vorkommen. Als Betriebssysteme wurden in der Microsoft Welt Windows 7 und in der UNIX Welt CentOS ausgewählt. Für die eigentliche Analyse wurde ein Shellskript erstellt, welches alle Dateien auflistet, diese danach zusammenfasst und nach Häufigkeit in Prozent sortiert. Das Shellskript kann unter UNIX ganz normal in der Shell und unter Windows mit Hilfe einer klassischen UNIX-Boot CD (z.B: Backtrack) ausgeführt werden, wobei natürlich vorher noch die Windows Partition gemountet werden muss. Das Skript besteht aus folgendem Code:

```
#!/bin/bash

find $1 -type f > tmp

#Dateipfade wegfiltern
sed 's/\\/.*\///' tmp > tmp2

#Dateien ohne Endung durch Pattern ersetzen
sed 's/[^.]*/FILE.NO_EXTENSION/' tmp2 > tmp

#Dateinamen ausfiltern
sed 's/.*\./' tmp > tmp2

count=1
oldfile="" _"
counter=$(wc -w tmp2 | cut -f1 -d' ')

for file in $(sort tmp2)
do
```

```
    if [ $oldfile == $file ]
    then
        let count=$count+1
    else
        count=$(echo "scale=4;\
.....$count/$counter*100" | bc -l)
        echo $count $oldfile >> tmp3
        oldfile=$file
        count=1
    fi
done

sort -g -r tmp3 > verteilung
rm tmp tmp2 tmp3
```

Listing 1: Der Quelltext des Analyse Skripts

3.3.1 Ergebnisse

—Windows 7—			—CentOS—		
Dateiendung	%	Beschreibung	Dateiendung	%	Beschreibung
.manifest	11,7	XML Daten	keine	23,8	Unbestimmt
.dll	10,21	Maschinencode	.png	14,07	Bilddaten
.lua	8,71	Quelltext	.gz	8,57	Komprimierte Daten
.mui	5,86	Text	.o	7,71	Maschinencode
.gif	4,74	Bilddatei	.mo	6,18	Quelltext
.jpg	3,53	Bilddatei	.xml	4,06	XML Daten
.png	2,80	Bilddatei	.html	3,12	XML Daten
.xml	2,59	XML Daten	.so	3,10	Maschinencode
.DLL	2,08	Maschinencode	.svgz	2,36	Bilddaten
.svn-base	2,07	Textdatei	.desktop	1,69	Text/Quelltext
.GPD	1,98	Quelltext	.py	1,47	Quelltext
.exe	1,95	Maschinencode	.la	1,37	Maschinencode
.php	1,76	Quelltext	.docbook	1,31	XML

Anhand dieser Verteilung sieht man, dass es ganz abstrakt drei verschiedene Typen gibt, die es zu klassifizieren gilt: Maschinencode (z.B.: ausführbare Programme), Textdateien(z.B.: XML Dateien, Quelltext,...) und Multimedia Dateien (z.B.: Bilder, Videos,...). Daher wird sich diese Arbeit auf diese drei abstrakten Dateitypen spezialisieren.

3.4 Data Mining Ansätze für verschiedene Dateitypen

3.4.1 Multimediadaten - Bilddateien

Bei Bildern ergibt sich der Nachteil, dass es sehr viele verschiedene Formate gibt. Die Populärsten in der Windows Welt sind laut der oben durchgeführten Analyse das JPEG und das GIF Format. Danach rangieren PNG und andere Formate(z.B.: TIF, BMP). Im Folgenden soll erläutert werden ob diese Formate individuelle Merkmale besitzen, die für die Data Mining Prozedur hilfreich sein könnten. Generell ist jedoch das Data Mining bei Bilddateien sehr schwer, da der Inhalt quasi komplett zufällig ist. Daher kann in den meisten Fällen nur der Sektor eindeutig als Bilddatensektor klassifiziert werden, der die ersten paar Bytes der Bilddatei beinhaltet. Dieser enthält nämlich in den meisten Fällen den Header, der eine eindeutige Zuordnung ermöglicht. Es ist jedoch unumgänglich auch Bilddateien als solche zu klassifizieren, da sie doch einen großen Teil des Dateisystems füllen. Zusätzlich ist aus der Sicht der Performance nicht schwer ein Bild durch Mustererkennung zu klassifizieren.

JPEG

Das JPEG Format kann neben den eigentlichen Bilddaten, die wegen der Zufälligkeit und der Komprimierung nicht sehr leicht klassifizierbar sind auch andere Daten (vgl. Hamilton 1992, S.2) enthalten. Diese sind in den sogenannten Segmenten enthalten. Dazu definiert das JPEG Format verschiedene Steuerzeichen, die eine bestimmte Art von Segmenten einleiten. Beispiele für Segmente wären Copyright Angaben, Kommentarfelder oder Datumsangaben. Am Anfang findet sich ein spezifischer String, der in den allen JPEG Bildern vorkommt (vgl. Hamilton 1992, S.2). Gängige Grafikbearbeitungsprogramme bzw. Software zum Betrachten von Bildern können zwar auch JPEG Bilder ohne diesen Header begutachten, jedoch ist dieser Header im Standard fest vorgesehen. In Abbildung 1 sieht man den JFIF Header bei zwei JPEG Bildern, die in einem Editor geöffnet wurden.

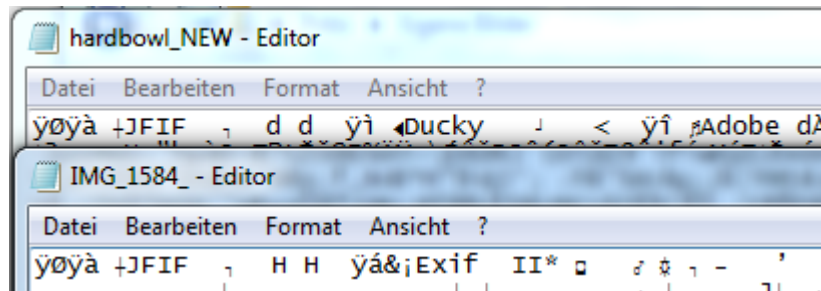


Abbildung 1: Der JFIF Header in zwei Bilddateien

Die Inhalte der einzelnen Segmente können ebenfalls Rückschlüsse auf den Datentyp lassen. Adobe Photoshop verewigt sich zum Beispiel beim Abspeichern eines Bildes durch eine Signatur fest in einem Segment (siehe Abbildung 2)

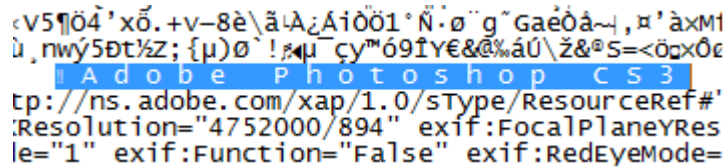


Abbildung 2: Signatur von Adobe Photoshop

Wie oben erwähnt bietet JPEG ebenfalls die Möglichkeit Metadaten mitzuspeichern (z.B. im EXIF Format). Diese Metadaten beinhalten grafikverwandte Ausdrücke, wie zum Beispiel das Farbmodell RGB, daher kann hier auch möglicherweise auf eine Bilddatei rückgeschlossen werden. Folgende Strings könnten dabei als Metadaten enthalten sein:

- ".IEC 61966-2.1 Default RGB colour space - sRGB"
- "xap:CreatorTool="Adobe Photoshop CS3 Windows" "
- "exif:ColorSpace="1""
- "exif:PixelYDimension="3012""
- "dc:format="image/jpeg""

- "photoshop:ICCProfile="sRGB IEC61966-2.1""
- "tiff:Model="Canon EOS 500D""

Diese Liste ist keinesfalls als komplett anzusehen, da es sehr viele verschiedene Parameter gibt, die man in einem Segment abspeichern kann, um im JFIF möglichst viel Meta-Information zu verewigen.

GIF

Beim GIF ist es laut dem Standard nicht möglich zusätzliche Metadaten zu speichern. Aufgrund dessen ist eine eindeutige Klassifizierung wie beim JPG Format nicht so leicht durchzuführen. Lediglich die sogenannte "Magische Zahl" lässt einen Rückschluss auf eine GIF Grafik zu. Dieser String ist am Anfang der Datei zu finden und hat den Inhalt "GIF87a" (vgl. CompuServe Incorporated 1987, S.4) oder "GIF89a" (vgl. CompuServe Incorporated 1990, S.7) (siehe Abbildung 3).

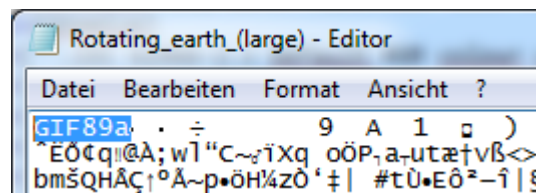


Abbildung 3: Magische Zahl in einer GIF Grafik

PNG

Bilddateien, die im PNG Format abgespeichert sind, lassen sich wie das GIF Format nur im ersten Sektor eindeutig identifizieren. Die ersten 8 Bytes beinhalten immer folgende dezimalen Werte "137 80 78 71 13 10 26 10" (vgl. Adler 1999, o.S.). Diese stellen die Signatur für das PNG Format dar (siehe Abbildung 4).

BMP

Das Bitmap Format hat ebenfalls einen Header, der erkannt werden kann. Dieser findet sich wieder wie gewohnt am Anfang der Datei und beinhaltet

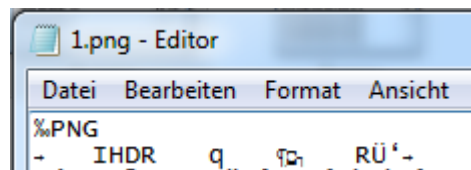


Abbildung 4: Die Signatur einer PNG Bilddatei

den kurzen String "BM" (vgl. FileFormat.Info o.J., o.S.). Daher lässt sich auch hier anhand dieses Strings keine sinnvolle Klassifizierung vornehmen, da dieser sehr kurz ist und daher auch in anderen Dateien zufällig vorkommen kann.

TIFF

Beim TIFF findet sich ebenfalls ein 8 Byte langer Header (vgl. Adobe Developers Association 1992, S.13), der jedoch nur 4 Byte an "statischer" Signatur enthält (entweder II42 oder MM42). Die restlichen 4 Byte enthalten den Offset zum ersten IFD (Image File Directory), der natürlich für jedes Bild anders sein kann. Was jedoch beim TIFF auffällt, ist das ebenfalls Metadaten (vgl. Adobe Developers Association 1992, S.28-41) wie beim JPG Format in den Bilddaten vorkommen (siehe Abbildung 5).

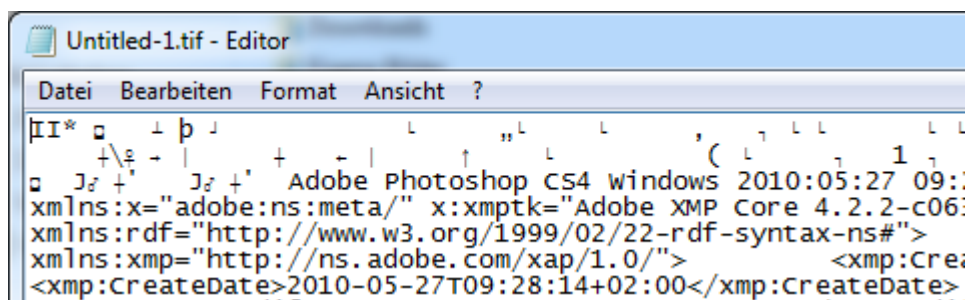


Abbildung 5: Der Inhalt einer TIFF Datei

3.4.2 Ausführbare Dateien

Ausführbare Dateien beinhalten Maschinencode, daher sollte ein statistischer Data Mining Ansatz möglich sein. Man muss hier zum Beispiel nur wissen, wie oft genutzte Maschinencode Befehle in das File codiert werden. Danach kann man nach diesen Pattern suchen und sie in Relation zu den Restlichen stellen. Falls die codierten oft genutzten HEX Pattern öfter vorkommen als die restlichen Pattern ist es daher sehr wahrscheinlich, dass dies eine ausführbare Datei ist. Aufpassen muss man hierbei auf die verschiedenen Prozessorarchitekturen und Betriebssysteme. Die folgende Analyse wurde auf einem Windows 7 x64 durchgeführt, wobei nur 32 Bit Programme verwendet wurden.

Analyse

Um Strukturen für Data Mining Prozeduren zu finden wurden hier auf ein Hilfsprogramm zurückgegriffen. Es nennt sich Crypt Tool und ist unter ¹ zum Download verfügbar. Es verfügt über die nützliche Analysemethode "Histogramm", die man hier gut gebrauchen kann.

Falls man nun dem Histogramm eine .exe oder .dll Datei als Input gibt, bekommt man eine Grafik angezeigt, die die Aufteilung der Bytes in dieser Datei anzeigt (siehe Abbildung 6).

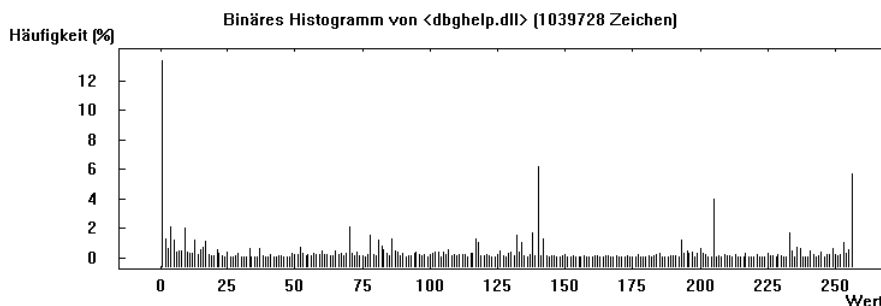


Abbildung 6: Das Histogramm einer ausführbaren Datei

Ein Histogramm wurde mit verschiedenen Dateitypen erstellt, damit man allgemeine Eigenheiten extrahieren kann. Dabei fiel auf, dass folgende HEX-

¹<http://www.cryptool.de/>

3 DATA MINING

Codes öfter vorkommen (siehe Abbildung 7):

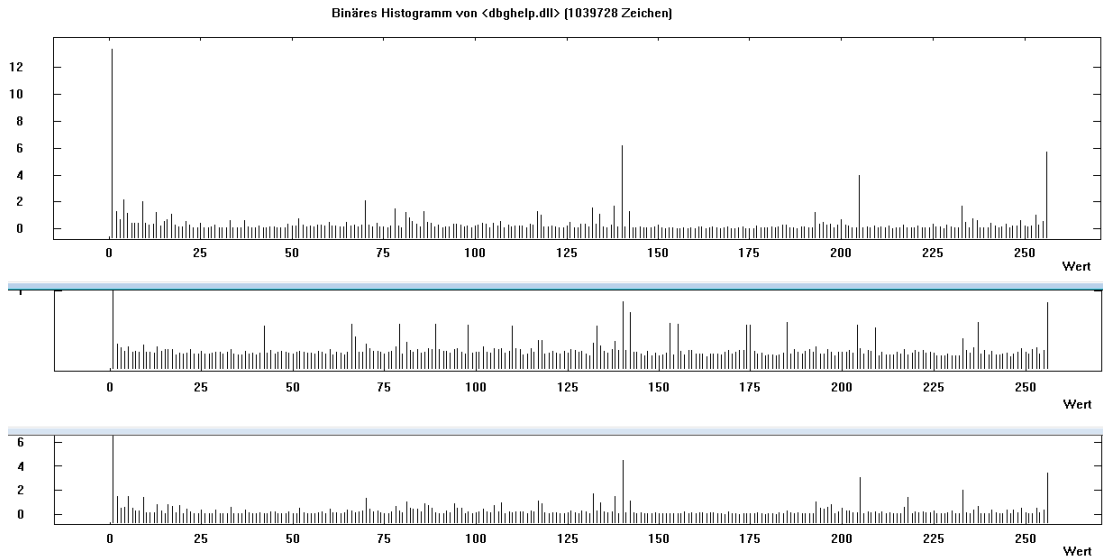


Abbildung 7: Vergleich der Histogramme

Bedeutung der Befehle (vgl. Ludloff o.J, o.S.) :

- **0x00:** ADD - Addieren
- **0xFF:** PUSH - Schreibt einen Wert auf den Stack
- **0x8C:** MOV - Schreibt einen Wert in ein Register
- **0xCD:** INT - Interrupt Procedure Call
- **0xE9:** JMP - Springt zur angegebenen Adresse

Wenn man beim Data Mining auf diese Verteilung Acht gibt, ermöglicht dies einen Data Mining Ansatz für Windows Executeables.

3.4.3 Texterkennung

Für die Erkennung von Text ist ein kleiner Exkurs über die Informationstheorie notwendig. Ein wichtiges Werk dazu ist "A Mathematical Theory of Information" von Claude E. Shannon. Ein Kommunikationssystem (vgl. Shannon 1948, S.2) besteht dabei laut Shannon aus folgenden Teilen :

- *Information Source*: Diese sendet Informationen an den Empfänger. Die Information kann dabei stark variieren, wie z.B. eine Folge von Zeichen (Telegraph) oder eine von der Zeit abhängige Funktion $f(t)$ (Radio).
- *Transmitter*: Die Aufgabe des Transmitters ist die Aufbereitung des Signals für die Übertragung über den Channel.
- *Channel*: Der Channel hat die Aufgabe die Information vom Transmitter zum Receiver zu übertragen.
- *Receiver*: Der Receiver übernimmt die inverse Funktion des Transmitters, d.h. eine Wiederherstellung des veränderten bzw. aufbereiteten Signals.
- *Destination*: Die Destination ist der Empfänger, für den die Nachricht bestimmt ist.

Shannon versuchte nun aufbauend auf dieses Modell mathematische Formeln zu definieren, die Information bzw. dessen Austausch in der Theorie beschreiben können.

Entropie

Eine wichtige Formel, die Shannon aufstellte, war die Formel für die Entropie (vgl. Shannon 1948, S.13) in der Informationstechnik. Sie beschreibt einfach ausgedrückt den mittleren Informationsgehalt eines Zeichensystems, dass von einer Quelle produziert wird. Die Quelle soll dabei laut Shannon folgende Eigenschaft besitzen: "We may consider a discrete source, therefore, to be represented by a stochastic process." (Shannon 1948, S.5). Unter einem *stochastic Process* kann man sich also eine einfache Zufallsvariable vorstellen.

Berechnen lässt sich die Entropie H mit der Information I in einem endlichen Alphabet Z mit folgender Formel (vgl. Shannon 1948, S.11):

$$H(I) = - \sum_{j=1}^{|Z|} p_j * \log_2 p_j$$

Ein ganzer Text kann somit auf den Informationsgehalt überprüft werden. Mit dem Programm CryptTool lässt sich ebenfalls wieder die Entropie für eine bestimmte Menge an Daten berechnen. Dazu wurden mehrere übliche deutsche Texte von verschiedenen Autoren auf die Entropie hin überprüft. Dabei ergaben sich bei einem eingestellten deutschen Alphabet von 26 Buchstaben folgende Entropiewerte:

Webseite	Entropie
http://de.wikipedia.org/wiki/Medizin	4,07
http://de.wikipedia.org/wiki/Internet	4,07
http://www.bildungsservice.at/nigg/dt5/beispieltext_1.htm	4,05
http://de.wikipedia.org/wiki/Text	4,03
http://de.wikipedia.org/wiki/Fachhochschule	4,11

Diese zeigen, dass man durch diese Werte klare Rückschlüsse auf einen deutschen Text ziehen könnte. Andere Dateitypen, wie zum Beispiel ausführbare Dateien oder Bilddateien erzielten bei der Recherche einen Entropiewert von 6,3 bzw. 7,93. Somit sollte sich ein Entropiewert für einen deutschen Text auf ca. 4,07 einpendeln, wobei man anmerken muss, dass diese Dateien alle größer als 512 Byte (die Sektorgröße bei Festplatten) sind. In der Praxis muss jedoch nachgeprüft werden, ob sich der Entropiewert bereits bei 512 Byte Text auf diesen Werten einpendelt und welche Standardabweichung optimal ist.

XML Dateien

Wie man bei den Dateiverteilungen sieht ist ein großer Teil der Daten XML. Diese ist mittlerweile vielen Betriebssystem und Anwendungen weit verbreitet um Daten getrennt von ihrer Darstellung abzulegen. Es besitzt eine standardisierte und strikte Syntax (vgl. W3C 2008, o.S.), die man sich zu Nutze machen kann.

- **Tags**

XML sowie auch HTML sieht eine Beschreibung der Daten durch sogenannte Tags (vgl. W3C 2008, o.S.) vor. Diese sind durch ein '<' am Anfang und ein '>' am Ende begrenzt. Daher sollte es optimalerweise genau bzw. annähernd soviele '<' wie '>' geben. Zusätzlich muss jedes Tag wieder von einem anderen Tag begrenzt werden. Diese hat die gleiche Syntax wie das ursprüngliche Tag, jedoch befindet sich nach dem öffnendem '<' ein Slash. Daher sollte statistisch gesehen für je zwei '<' ein '/' im Text existieren.

- **Header**

XML besitzt einen eigenen Header (vgl. W3C 2008, o.S.), der jedoch nur am Anfang steht. Sollte dieser jedoch erkannt werden ist es ein sicherer Beweis für ein XML Dokument.

4 Implementierung

Für eine mögliche Implementierung der obigen Erkenntnisse in einem Programm möchte ich noch ein paar Anmerkungen hinzufügen:

- **Festplattenzugriff:**

Der Zugriff auf die Daten der Festplatte muss RAW erfolgen, d.h. ohne das Dateisystem als Zwischenschicht. Dabei werden wirklich alle Daten Bit für Bit aus den jeweiligen Festplattensektoren gelesen. Besonders muss darauf geachtet werden, dass die Datenmenge, die ausgelesen wird, ein Vielfaches der Sektorengröße sein muss (512 Byte, 1024 Byte, etc...). Dies ist jedoch von Dateisystem zu Dateisystem unterschiedlich.

- **Abbildung von Unsicherheit**

Bei der anschließenden grafischen Darstellung könnte die Unsicherheit, die bei der Analyse entsteht, abgebildet werden. Man stelle sich zum Beispiel das Szenario vor, dass ein Sektor mit annähernd gleicher Wahrscheinlichkeit zwei verschiedenen Klassen zugeordnet wird. Somit könnten bei der nachfolgenden grafischen Einsicht der Ergebnisse durch die Abbildung dieser Unsicherheit eventuelle Missverständnisse vermieden werden.

- **Suchalgorithmus**

Da ein Großteil der Performance des Programms vom verwendeten Suchalgorithmus abhängt, muss dieser optimiert werden, um bei vielen Suchschritten eine gute Performance zu gewährleisten. Der Suchalgorithmus in diesem Programm besteht aus zwei großen Teilen, nämlich der Mustererkennung und der statistischen Analyse. Da die Mustererkennung sehr schnell ist, da diese auf der schnellen Suche nach Strings basiert, wird diese vor der statistischen Analyse durchgeführt. Liefert die Mustererkennung keine Erfolgsmeldung zurück, wird auf die statistische Analyse umgeschwenkt. Wenn diese kein Ergebnis liefert, kann keine Aussage über den Sektortyp getroffen werden. Unter der groben Ebene der Mustererkennung und statistischer Analyse kann erneut eine Differenzierung erfolgen. Es kann zum Beispiel wenn ein Sektor als Textdatei identifiziert wird, nochmals eine Überprüfung auf XML oder Quelltext erfolgen (siehe Abbildung 8).

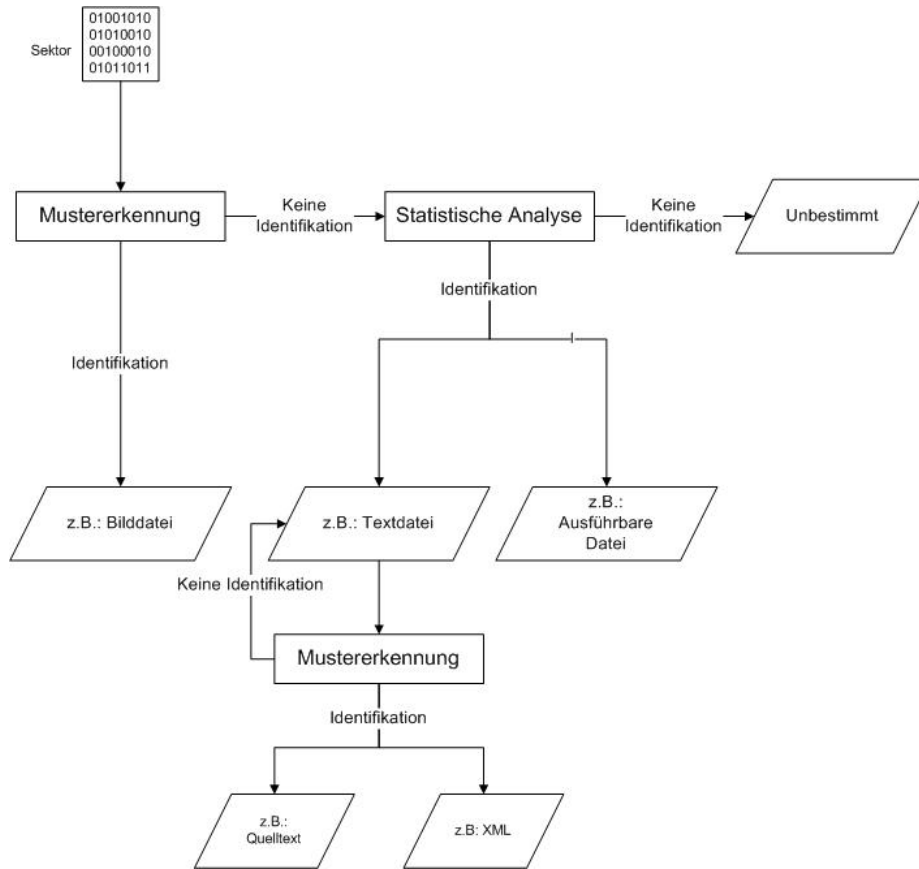


Abbildung 8: Flussdiagramm des Suchalgorithmus

Wie man dem Diagramm entnehmen kann verwendet dieses Konzept den Ansatz des konzeptionellen Clusters. Dabei werden die einzelnen Cluster immer weiter unterteilt bzw. spezialisiert. Wenn die Abbildung der Unsicherheit, wie oben erwähnt, umgesetzt wird kann der Cluster Ansatz auch auf den Bayes Ansatz erweitert werden.

5 Schlussfolgerung und Ausblick

Es regt zum Nachdenken an, dass es keine frei verfügbaren wissenschaftlichen Arbeiten bezüglich der Klassifizierung von Daten gibt. Dies liegt wahrscheinlich daran, dass jeder Entwickler, der solche Methoden entwickelt, diese danach nur gegen Geld oder als ganzes Programm ohne Quellcode verfügbar macht. Es ist jedoch in den meisten Fällen gar nicht komplex, den zu Grunde liegenden Algorithmus für die Klassifizierung bestimmter Dateitypen zu finden. Es wurde zum Beispiel im Rahmen dieser Arbeit herausgefunden, dass in einer ausführbare Datei bestimmte Bytes öfter vorkommen, da manche Assemblerbefehle statistisch gesehen öfter als andere vorkommen. Auch die Erkennung von Text ist wegen der konstanten Verteilung von Buchstaben in jeder Sprache mit Hilfe der Entropie möglich. Die Klassifizierung mit Hilfe von Pattern (z.B. Header) ist in dieser Arbeit eher Nebensache, da diese weniger innovativ ist.

Aufbauend auf diese Arbeit wäre ein Programm wünschenswert, das die oben erwähnten Erkenntnisse umsetzt. Zusätzlich wäre eine genauere Erkennung der Datentypen bzw. eine weitere Differenzierung wünschenswert. Man könnte beispielsweise die Textdateien weiter in Quelltextdateien der einzelnen Programmiersprachen aufspalten. Auch eine Optimierung des im Rahmen der Arbeit entwickelten Suchalgorithmus würde sich positiv auf dem Fortschritt in diesem Gebiet auswirken.

6 Quellenverzeichnis

Adobe Developers Association. (1992). TIFF. [<http://partners.adobe.com/public/developer/en/tiff/TIFF6.pdf>] (ausgehoben 22.06.2010)]

Alder, M. u.a. (1999). PNG (Portable Network Graphics) Specification. [<http://www.mirrorservice.org/sites/www.libpng.org/pub/png/spec/1.2/PNG-Structure.html>] (ausgehoben 22.06.2010)]

Bissantz, N. / Hagedorn, J. (2009). Data Mining (Datenmustererkennung). In: Wirtschaftsinformatik Vol. 51 Nr. 1 vom Februar 2009, S. 139-144. [<http://www.springerlink.com/content/k023w98876215k43/>] (ausgehoben 16.04.2010)]

CompuServe Incorporated. (1987). Graphics Interchange Format. [<http://www.w3.org/Graphics/GIF/spec-gif87.txt>] (ausgehoben 22.06.2010)]

CompuServe Incorporated. (1990). Graphics Interchange Format. [<http://www.w3.org/Graphics/GIF/spec-gif89a.txt>] (ausgehoben 22.06.2010)]

FileFormat.Info. (o.J.). Microsoft Windows Bitmap Format. [<http://www.fileformat.info/format/bmp/egff.htm>] (ausgehoben 22.06.2010)]

Hamilton, E. (1992). JPEG File Interchange Format. [<http://www.jpeg.org/public/jfif.pdf>] (ausgehoben 22.06.2010)]

Ludloff, C. o.J. IA-32 architecture one byte opcodes. [http://www.sandpile.org/ia32/opc_1.htm] (ausgehoben 22.06.2010)]

Shannon, C. E. (1948). A Mathematical Theory of Communication. [<http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf> (ausgehoben 29.05.2010)]

W3C. (2008). Extensible Markup Language (XML) 1.0 (Fifth Edition). [<http://www.w3.org/TR/REC-xml/> (ausgehoben 22.06.2010)]

7 Abbildungsverzeichnis

Abbildungsverzeichnis

1	Der JFIF Header in zwei Bilddateien	13
2	Signatur von Adobe Photoshop	13
3	Magische Zahl in einer GIF Grafik	14
4	Die Signatur einer PNG Bilddatei	15
5	Der Inhalt einer TIFF Datei	15
6	Das Histogramm einer ausführbaren Datei	16
7	Vergleich der Histogramme	17
8	Flussdiagramm des Suchalgorithmus	22