Document Type Detection

Agenda

- Motivation for Automated File Detection
- Use Cases
- File Formats
- Approaches for Automated File Detection
- BFD, BFC, and FHT
- Datasets to use for Document Type Detection
- Future Work

Why automated file detection

- **Virus Scanning**
  - Many virus scanning software tools rely on the ability to identify which files may contain viruses in them, and so the ability to automatically detect executable files is key
- **Firewalls / Networking / Intrusion Detection Systems**
  - At the network level, allowing e.g., SMTP communication on port 25 with an EXE attachment or Python file is important to detect
- **File Forensics**
  - When handed a bunch of disks with data on them, being ability to accurately classify that data for legal purposes, etc., is key
- **Scientific Data / Processing / Big Data** about common cloud
  - Sorting level 1 data (raw telemetry) from processed geo-referenced science data (level 2)

More Use Cases

- **Content Repositories**
  - Document Sharing systems, e.g., share point need to automatically identify file types – put files accessed more often (e.g., News such as RSS) on faster servers, rather than less frequently changed (e.g., PDFs)
- **Web Crawling**
  - Go after certain types of files for your e.g., food rating review system (e.g., Microformats, Microdata), ignore other file types

Digital Cyber Investigations

file can be intentionally spoofed and look innocuous when it's not

- File extensions and even MIME magic is no longer enough to detect file types!
- Forensic investigators have found critical evidence (images, multimedia, etc.) embedded in otherwise "safe" file types (Alamri & Allen, IEEE SouthEastCon 2015)
- Extensions are often changed to "safe" file types and encoded with bad content (viruses)

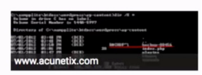
Some History in the Space

- **Operating Systems**
  - Need to associate applications to open with file types
  - As far back as the DEC operating system
  - Original filenames in File Access Table (FAT) filesystem limited to 8 character letters and 3 char extension (8.3 filename)
- **Virus Scanners**
  - Default in my cases to scanning only executable files, so understanding which files are executable and having a high degree of accuracy and rapid response in that detection is necessary – could mean the difference in detecting Virus or not.

8.3 filenames

8 characters . 3 characters – extension.

An **8.3 filename** (also called a short **filename** or SFN) is a **filename** convention used by old versions of DOS and versions of Microsoft Windows prior to Windows 95 and Windows NT 3.5. It is also used in modern Microsoft operating systems as an alternate **filename** to the long **filename** for compatibility with legacy programs.



8.3 filename - Wikipedia
https://en.wikipedia.org/wiki/8.3_filename

File Formats

language of how to read characters

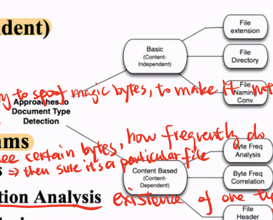
- "A file format is a standard way that information is encoded for storage in a computer file. It specifies how bits are used to encode information in a digital storage medium. File formats may be either proprietary or free and may be either unpublished or open."
- https://en.wikipedia.org/wiki/File_format

File Formats

- **Free file formats**
  - Difference between e.g., Open Office (.odf) and e.g., Microsoft Office (.docx)
  - Specification for file header, encoding (compression), etc., provided openly so as to allow many reference implementations
    - LibreOffice, Apache OpenOffice, Apache Open Document Format (ODF) Incubating
- **Closed file formats**
  - Protected by trade secrets, non disclosure agreements, specifications aren't public
  - Can be reverse engineered using techniques like we will discuss today
  - File format patents are not directly permitted under US law, some file formats encode data using patented algorithms but can apply it to encode data
  - Big decline in GIF because of this (compression), and PNG increase

Approaches for automatic detection

- **Basic (Content-Independent)**
  - File extension
  - File Directory Structure
  - File Naming Convention — try to spoof magic bytes, to make it not obvious what kind of file it is
- **Content-Based Algorithms**
  - Byte Frequency Analysis — see certain bytes, how frequently do certain bytes occur ⇒ then sure it's a particular file; eg: HTML ⇒ left cares and right cares ← b1 < b2 bytes are seen often
  - Byte Frequency Correlation Analysis — existence of one type of byte usually implies existence of another
  - File Header Trailer Analysis — don't have to look through full file, just look at the top and the bottom; eg: magic bytes at beginning helps to detect
  - /etc/magic



Basic approaches

- 1. **File Extension based** – can easily be spoofed. Microsoft's Operating Systems typically have used this approach almost exclusively. OS maintains table of associations and must be manually updated by user.
- 2. **/etc/magic file** – Special file in UNIX oriented systems that look at first 16 bits of each file and then associates these as a magic number with each file. Table is maintained in /etc/magic file
- 3. **Container File Formats** – encapsulate other files

Some limitations of /etc/magic

1. Magic numbers must be predefined before the files are generated
2. Makes it difficult to evolve and adapt magic numbers over time since it would require having to go back and adapt these signatures in existing files
3. Not all file types use magic numbers (we'll see more about this later)
4. Since only 16-bits allocated, had to expand to handle #! special operator later (indicates other commands to run on the file and that file is script)

limited space

Container file formats

- **Standard Archive Format (SAF)**
  - http://www.fileformat.info/format/saf/egff.htm
  - Archival storage of multiple data types in a standard format
  - Proprietary

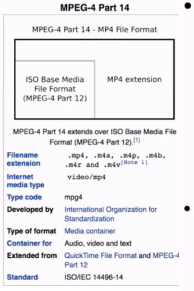
```
HdSize Auto
Class  Unclassifie
DaType Int8
Keywrd CMAP
XPixls 6
YPixls 6
ComPrs NONE
Data
ff ff ff ab 0f 0d.
```

- **OLE (Object Linking and Embedding Compound File)**
  - http://www.forensicswiki.org/wiki/OLE_Compound_File
  - Originally Proprietary (Microsoft), became part of Open Office
- Formats include special information that indicate MIME type of internal files
- A bigger list is here: https://en.wikipedia.org/wiki/Digital_container_format

Another container file format

maybe this special extension

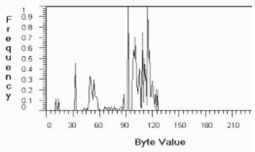
- MPEG Layer 4
- Nice elements
  - Open standard (ISO)
  - Many proprietary implementations initially didn't all play the same way
  - Has largely been standardized now



https://en.wikipedia.org/wiki/MPEG-4_Part_14

Content-Based File Type Detection

- **McDaniel & Heydari, 2003**
  - Investigated three approaches
  1. Byte Frequency Analysis **(BFA)**
  2. Byte Frequency Correlation **(BFC)**
  3. File Header Trailer **(FHT)**



- **Some key targets**
  - **Accuracy** – approach should be highly accurate (overcome filenames) and difficult to spoof (security/firewalls/viruses)
  - **Automatic** – lots of files, big data, need to run fast
    - Fingerprints should be small and easily comparable
  - **Flexibility** – don't want to have to maintain tables

Format of /etc/magic file

- https://www.mkssoftware.com/docs/man4/magic.4.asp - file tool in UNIX
- Lines contain byte sequences that can be compared
- ">" indicates a continuation of the previous line and any patterns preceded by this character are checked
- "&" indicates continuation of previous line and MUST match this byte sequence.
- First col is the byte offset, then data type, then byte sequence to compare and then descriptive name for the type

EXAMPLES

Here are some sample entries:

```
0       byte    0x80                    OMF object file (Microsoft relocatable)
0       short   0x5A4D                  extended DOS executable (.EXE)
>*0x3C  short   0x454C                  -- Windows VxD Linear format
>*0x3C  short   0x4550                  -- Win32 or NT Portable format
>>54    byte    2                       -- Windows format
>0x32   string  PKWARE                  Self extracting Zip
>0x24   string  LHA's                   Self extracting LHA
>0xE9   string  PKSFX2                  Self extracting Zip
0       short   xFEFF                   Windows Registry exported text file (.REG)
&2      Ustring                         Windows Registry Editor Version 5.00
```

Ongoing research regarding MAGIC

- Research by Beebe, Maddox et al. 2013 on using unigrams (single bytes) and bigrams (byte pairs) to create n-grams (groups of bytes)
- General research area is deriving a representative *fingerprint* (recall Deduplication lecture)
- Most modern approaches are focused on *byte frequencies* (more on this later)

Container file format limitations

1. Many began as Proprietary – vendor lock in
2. Brings together data and metadata – usually better to let them evolve separately
3. Wrappers are necessary and we've created another level of indirection to get at the internal constituent files.
   - And a new set of Magic, and need for detecting

Of course we have seen an example of these: message/* and multipart/* attachments from top level types of MIME taxonomy

Learning from the actual File Data

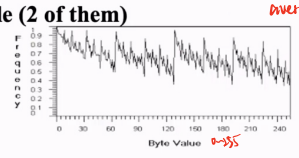
- Previous approaches were somewhat limited
  1. Rely on filenames which are easily spoofed and/or changed
  2. Rely on tables that are difficult to discern and must be maintained and updated (Magic)
  3. Rely on new proprietary "wrapper formats" (Containers)
- **Key Insight:** What if we simply examined the data inside many existing files and tried to develop a "fingerprint" a la Similarity metrics to identify files => Content-based approaches

identify through pattern

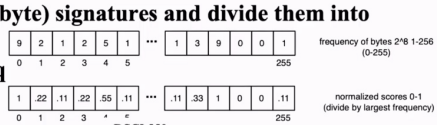
Byte Frequency Distribution (BFD)

- GIF file (2 of them)

average distribution → fingerprint → pattern

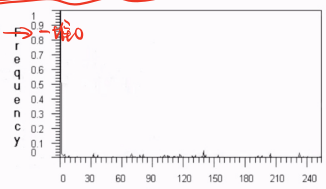


- Use 8-bit (byte) signatures and divide them into bins
  Count freq in each



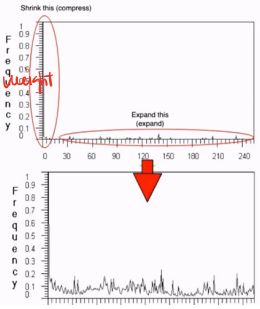
Some issues with BFD

- BFD for an EXE file
- Note it has a large variation between 0 byte and small signatures in rest
- Use a companding function to amplify the smaller frequencies
- Smaller frequencies lead to difficulty in BFD signature accuracy and efficacy



Companding process

- Companding amplifies low signal and compresses high signal
- Results after companding of the ~~GIF~~ file
- Since original numbers were normalized 0..1 the result companding will be too



Companding function

- https://www.techopedia.com/definition/24121/companding
- From the signal processing domain
- "Compressing" and "Expanding" = companding
- From Wikipedia https://en.wikipedia.org/wiki/Companding
  - "mitigating the detrimental effects of a channel with limited dynamic range"
  - In our case, mitigating limited dynamic range of frequencies by compression and expanding

Finalizing BFDs

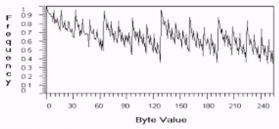
- Resulting BFDs are generated for file types based on the actual contents of the file – somewhat limiting b/c you have to scan the whole file for traces of the byte fingerprints
- Each file BFD is then averaged across many samples from the same type
  - Take 100 GIFs and generate BFD and then average them
- Resulting averaged BFD is a now the generalized indicator or *fingerprint* for a file type

BFD "correlation" strength

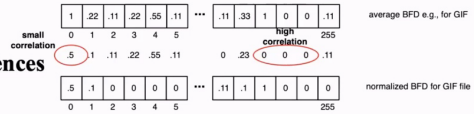
- If a byte value occurs with some regular frequency f in a file type then it is an important feature by definition for file identification
- Compute "correlation strength" between a type (e.g., GIF's) BFD and a particular file type
- Difference the BFD from an input file and its overall BFD fingerprint
  - Compute a "similarity" or "correlation" between an input file and its BFD
  - If frequency is close to fingerprint, then file is likely, e.g., a GIF

BFD Correlation

- GIF files

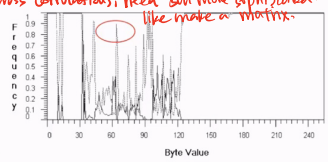


- "correlation": ↑ frequency distributions match
- Could use sophisticated techniques like KS-test, D_KL, etc.,
- Average BFD fingerprint
- Corr differences



Byte Frequency Cross Correlation (BFC)

- In looking at Byte Frequency Correlation, sometimes it is possible to notice e.g., equal sized spikes between two byte markers (pair-wise) in a file type
- Example
  - HTML files, byte values 60 "<" and 62 ">"
- This is called "cross correlation"



Two key pieces of BFC

- Average difference in frequency between all byte pairs
- Correlation strength of any new file and that average frequency
- Cross-correlation array is built 0-255, to represent correlation between byte pairs

	0	1	...	255
0	nf	2	..	9
1	.22	nf	..	2
⋮	..	..	nf	..
255	1	.22	..	nf

diagonals can store info e.g., number of files that have been used to compute the fingerprint

half the matrix can be thrown out since it contains redundant information

(i,j) stores the freq of co-occurrence of i and j bytes (j,i) stores the normalized value

- Can be compared to new files

File/Header Trailer (FHT)

- Sometimes BFA and BFC are unable to identify file types b/c no distinct patterns
  - Text/plain can have such varied byte patterns, it's hard to specialize
- Also BFA/BFC has to full file scan
- File Headers and Trailers – patterns of bytes that appear at fixed location at beginning and end of file
  - Reduce full file scan
  - Still take into consideration strong byte frequencies (BF)
- Pick *H* header bytes from beginning file
- Pick *T* trailing bytes from end of file
- Generate *H* x 256 matrix: and *T* x 256 matrix

General process: FHT

1. Compute H and T matrices
2. Flag a 1 if the header byte is equal to the col value (0-255), 0 otherwise
3. Sparse matrix

	0	1	...	255
H byte 0	0	1	..	0
H byte 1	.1	0	..	0
⋮	..	..	0	..
H byte H	0	0	..	1

sparse matrix is a 1 if that header byte value (0-255) is present at byte 0, 0 otherwise

average and compute correlation strength across sample set

- FHT is strong motivation for MIME "magic"
- GIF specifies GIF87a; GIF89a at byte 0

Evaluation of BFD, BFC and FHT

- 30 file type fingerprints across 120 files
  - ACD, DOC, PPT, XLS
- BFA accuracy is only 27.5%
  - Better than random guess, but not by much
  - Speed was 0.010 seconds (fastest; file size may be a factor)
- BFC accuracy is only 45.83%
  - Speed was 1.19 seconds (worst)
- FHT accuracy is 95.85%
  - Also was 2nd place in speed, not far off (0.015 seconds per file)

The GovDocs dataset

- http://digitalcorpora.org/corpora/govdocs
- 1 million files
  - Made available in multiple ways: set of 1000 x 1000 directories
  - 1000 ZIP files
  - Tar files
  - A set of 1000 directories, with 1000 files in each directory, downloadable from our server at http://digitalcorpora.org/corp/files/govdocs1/.
  - Subsets of the data
- Metadata to go along with the files
  - URL, download time, search term, md5 hash, dublin core metadata (MIME type)

Digital Corpora
Producing the Digital Body

One of Garfinkel's famous experiments

- In 2003, Garfinkel and Abhi Shelat published an article in IEEE Security & Privacy Magazine reporting on an experiment in which they purchased 158 used hard drives from a variety of sources and checked to see whether they still contained readable data. Roughly one third of the drives appeared to have information that was highly confidential and should have been erased prior to the drive's resale.

Common Crawl Dataset

- http://commoncrawl.org/the-data/get-started/
- Data is available in ARC (compressed) file format
- 7 years and petabytes of crawled and collected data
- Made available as part of Amazon's Public Dataset program
  - http://aws.amazon.com/public-data-sets/

Tika and Document Type Detection

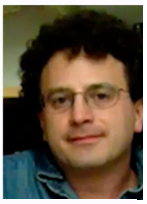
- Use GovDocs as a release-oriented test
  - Use Tika Batch (we'll discuss later) to iterate over GovDocs on a machine freely provided by RackSpace
  - Compare precision/recall and regressions against previously computed MIME types to test detectors
  - https://wiki.apache.org/tika/TikaBatchOverview
  - https://wiki.apache.org/tika/VirtualMachine

FHT promising approach with speed and accuracy

- Natural evolution is to MIME magic
  - Extend H and T signatures with priority / precedence
  - Simplify by treating H and T the same, simply with offsets
- Common approaches today (Tika does this)
  - File extension hint (allow strong override from MIME magic)
  - Content-Type hint (potentially if available from metadata surrounding file, e.g., if from web server)

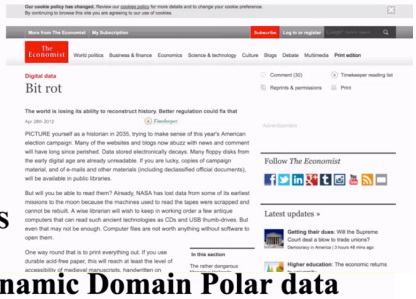
GovDocs Dataset

- Garfinkel, Farrell, Roussev and Dinolt, Bringing Science to Digital Forensics with Standardized Forensic Corpora, DFRWS 2009, Montreal, Canada
- Simson Leon Garfinkel
- Worked at NIST, and Naval Postgraduate School
- Expert in computer science forensics and computer security



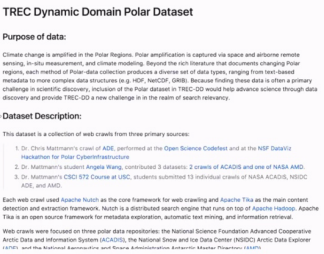
Bit rot

- How to preserve and assess digital data DECADES or CENTURIES later?



TREC Dynamic Domain Polar data

- 1.7M urls
- 93+ different web MIME types
- Collected across three of the most widely used NSF, NASA arctic data repositories from 2015-2016
- Data available on Amazon S3 and at Arcticdata.io



http://github.com/chrismattmann/trec-dd-polar/

Current Research Trends

- BFD and BFC rarely looked at for some of the modern scientific data file formats (and other increasing formats)
- Gridded Binary File Format and BFD/BFC
- GRIB is a popular scientific format
- Question: can we train Neural Networks to learn BFD for file types?

https://issues.apache.org/jira/browse/TIKA-1582

Modern File Type Identification Techniques

- Alamri and Allen, 2015
- Modern ML approaches
  - SVM classifier
  - K-nearest neighbor
  - Neural Networks
  - Linear Discriminant Analysis
- All relies upon what features to select
- How to "featurize" the file type?

Some recent results

- Highest accuracy is 97%
  - K-NN with 8 features 500 byte files
  - NN-MLP with 64 features and 500 byte files
- Feature size isn't dependent
- Sample size likely too low

TABLE III. PARAMETERS THAT PRODUCED THE BEST CLASSIFICATION RATE FOR THE LINEAR DISCRIMINANT ANALYSIS APPROACH

LDA	Number of Features	Fragment Size	Training Data Ratio	Classification Rate
50 files	64	500 byte	90/10	93%
100 files	64	500 byte	80/20	88%
50 files	64	1000 byte	70/30	91%
100 files	64	1000 byte	80/20	91%
50 files	64	2000 byte	70/30	80%
100 files	64	2000 byte	80/20	85%

TABLE IV. PARAMETERS THAT PRODUCED THE BEST CLASSIFICATION RATE FOR THE SUPPORT VECTOR MACHINE APPROACH

SVM	Number of Features	Fragment Size	Training Data Ratio	Classification Rate
50 files	64	500 byte	90/10	93%
100 files	64	500 byte	80/20	94%
50 files	16	1000 byte	90/10	93%
100 files	64	1000 byte	80/20	92%
50 files	16	2000 byte	90/10	90%
100 files	64	2000 byte	80/20	92%

TABLE V. PARAMETERS THAT PRODUCED THE BEST CLASSIFICATION RATE FOR THE K-NEAREST NEIGHBOR APPROACH

k-NN	Number of Features	Fragment Size	Training Data Ratio	Classification Rate
50 files	32	500 byte	80/20	90%
100 files	8	500 byte	90/10	97%
50 files	16	1000 byte	70/30	84%
100 files	12	1000 byte	90/10	95%
50 files	16	2000 byte	90/10	90%
100 files	12	2000 byte	70/30	89%

TABLE VI. PARAMETERS THAT PRODUCED THE BEST CLASSIFICATION RATE FOR THE NEURAL NETWORK (RBF) APPROACH

NN-RBF	Number of Features	Fragment Size	Training Data Ratio	Classification Rate
50 files	4	500 byte	80/20	82%
100 files	4	500 byte	80/20	87%
50 files	4	1000 byte	90/10	80%
100 files	4	1000 byte	80/20	88%
50 files	4	2000 byte	80/20	77%
100 files	4	2000 byte	80/20	88%

TABLE VII. PARAMETERS THAT PRODUCED THE BEST CLASSIFICATION RATE FOR THE NEURAL NETWORK (MLP) APPROACH

NN-MLP	Number of Features	Fragment Size	Training Data Ratio	Classification Rate
50 files	64	500 byte	90/10	97%
100 files	32	500 byte	70/30	92%
50 files	32	1000 byte	90/10	93%
100 files	64	1000 byte	80/20	94%
50 files	16	2000 byte	70/30	87%
100 files	32	2000 byte	80/20	91%

TABLE VIII. BEST RESULTS FOR EACH APPROACH

	Files	Number of Features	Fragment Size	Training Data Ratio	Classification Rate
LDA	50	64	500 byte	90/10	93%
SVM	100	64	500 byte	80/20	94%
k-NN	100	8	500 byte	90/10	97%
NN-RBF	100	4	1000 byte	80/20	88%
NN-MLP	50	64	500 byte	90/10	97%

Some Features

- Unigrams (single bytes)
- Bigrams (byte pairs)
- N-grams (groups of bytes)
  - Learn on these
- How to divide the file
  - 512 byte blocks (Sportiello et al show that an average of 28k blocks per file type in their studies)
  - 500, 1000, 2000, 8000 bytes, or whole file
- How to divide the number of files per type
  - 30, 50, 100

Parameter for Experiment	Range of parameters used
Number of features input to the classifier	2, 4, 8, 12, 16, 32, or 64
Number of files in the sample set used as input	30, 50, 100
Size (in bytes) of file fragment used as input	500, 1000, 2000, 8000, whole file
Ratio of training data to test data	50/50, 60/40, 70/30, 80/20, 90/10

Summary

- Important to detect document types
  - Operating System
  - Web Crawling
  - Anti-Virus and Security
  - Forensics
- Techniques center around non-content (basic) approaches and Content-Based approaches
- Content-based take precedence
  - Fast, Accurate, and based on the Data (don't require apriori knowledge)
- Evaluation of Content-based approaches
- Datasets