Chapter 17

# DATA TYPE CLASSIFICATION: HIERARCHICAL CLASS-TO-TYPE MODELING

Nicole Beebe, Lishu Liu and Minghe Sun

**Abstract** Data and file type classification research conducted over the past ten to fifteen years has been dominated by competing experiments that only vary the number of classes, types of classes, machine learning technique and input vector. There has been surprisingly little innovation on fundamental approaches to data and file type classification. This chapter focuses on the empirical testing of a hypothesized, two-level hierarchical classification model and the empirical derivation and testing of several alternative classification models. Comparative evaluations are conducted on ten classification models to identify a final winning, two-level classification model consisting of five classes and 52 lower-level data and file types. Experimental results demonstrate that the approach leads to very good class-level classification performance, improved classification performance for data and file types without high entropy (e.g., compressed and encrypted data) and reasonably-equivalent classification performance for high-entropy data and file types.

**Keywords:** Statistical classification, data types, file types, hierarchical model

## 1. Introduction

Statistical data type classification has many important applications in cyber security and digital forensics. Cyber security applications include intrusion detection, content-based firewall blocking, malware detection and analysis, and steganalysis. Data type classification can defend against many common signature obfuscation techniques and enhance the detection and blocking of undesired network traffic. It can also help map binary objects [12], which is useful in malware analysis and, possibly, steganalysis. In digital forensics, data type classification aids

fragment identification, isolation, recovery and file reassembly. Commercial and open-source tools such as `file` and TrID are reliant on file signatures and other magic numbers, rendering them ineffective when file headers and/or other blocks containing key magic numbers are missing or corrupted, or when their locations in the files are unknown [16]. Data type classification can also aid forensic triage efforts and improve investigative efficiency by targeting or prioritizing investigative efforts and search results [8].

This research focuses on data type classification absent reliable file signatures, filename extensions and other filesystem data that may identify the data type, either based on the file type that the data fragment used to be a part of in the case of files and composite objects, or based on the data type or primitive data type as defined by Erbacher and Mulholland [13]. It is important to note that, when reliable file signatures, filename extensions or filesystem data exist pertaining to a data fragment, traditional file signature based methods should be used over statistical or specialized [24] data type classification methods, including the hierarchical modeling approaches explored in this chapter. However, in instances where such reliable metadata and/or magic numbers do not exist, alternative data type classification methods are needed.

A fair amount of data type classification research has been conducted in the past decade, especially in the digital forensics domain. Researchers have explored a wide variety of data and file types, classification algorithms and feature sets. Several researchers have limited their multi-class size to ten or less [1–4, 6, 10, 11, 13, 18–21, 23, 26]. Fewer researchers have investigated data type classification methodologies for scenarios involving more than ten classes [9, 12, 14, 16, 17, 22, 25]. Overall, support vector machines (SVMs) have prevailed, achieving the best balance between prediction accuracy and scalability [9, 14, 16, 20]. Research has also demonstrated the great discriminatory value of $n$-grams across a wide range of data and file types [9].

An evaluation of the extant research reveals that one of the most negative influences on prediction accuracy is the number of classes in a multi-class problem. As the number of classes increases, prediction accuracy tends to decrease, as shown in Figure 1, which records empirical prediction accuracy relative to the number of classes reported in sixteen data type classification publications [1, 2, 4–6, 10–12, 14, 18, 19, 21–23, 25, 26]. Reducing the number of classes is highly advantageous. However, many data type classification use cases require the ability to classify blocks among a large number of classes. Therefore, this research empirically examines a hierarchical modeling approach proposed by Conti et al. [12]. The approach classifies data and file types first into classes and

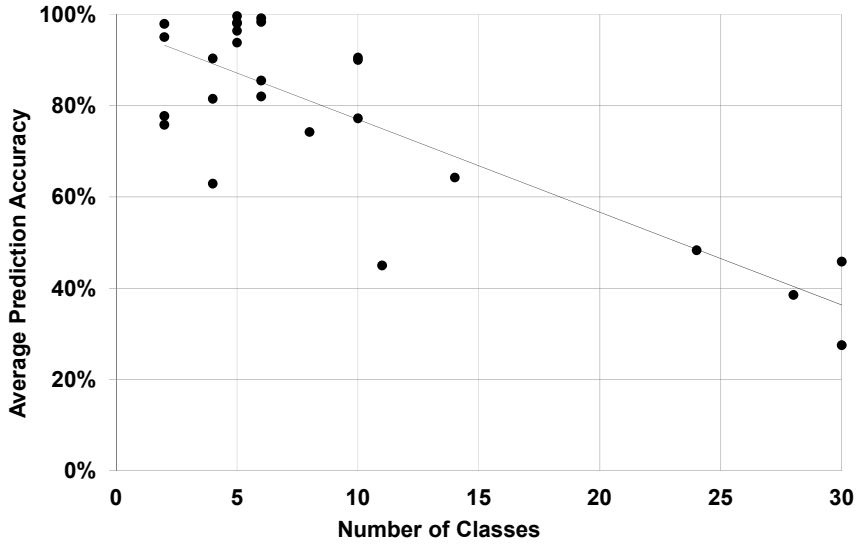


*Figure 1.* Average prediction accuracy vs. number of classes.

further classifies them into class-based types. This approach is expected to increase the overall prediction accuracy because each sequential classification problem becomes smaller in terms of the number of classes in each individual multi-class problem.

## 2. Methodology

This research employed a traditional empirical approach (hypothesis testing) as well as an exploratory approach. First, the six-class, two-level hierarchy shown in Table 1 was hypothesized based on knowledge of the internal data structures and encodings within each data and file type. Next, an exploratory data analysis was conducted to derive alternative hierarchical models from the data. Finally, a comparative evaluation was performed on the prediction accuracy at the class level and the type level. To accomplish this, two clustering algorithms, $k$-means and expectation maximization, were employed. The $k$-means algorithm was used to set the number of classes (clusters) *a priori* equal to the number of hypothesized classes; the model was considered to be validated if the resulting type-to-class cluster distribution matched the hypothesized hierarchical model. Expectation maximization clustering was used to derive alternative hierarchical models from the data. The precision, recall and overall classification accuracy levels were measured at the class level and file/data type level to determine the winning hierarchy. Two sample sizes ($n = 20$ and $n = 50$ samples) were employed per type and

*Table 1.* Hypothesized six-class, two-level hierarchy.

| **TEXT** | **PSEUDO-RANDOM** |
|---|---|
| Delimited (.csv) | Encrypted (AES256) |
| JSON (.json) | Random |
| Base64 (N/A) | |
| Base85 (N/A) | **COMPRESSED-LOSSY** |
| URL Encoding (N/A) | MP3 (.mp3) |
| Postscript (.ps) | AAC (.m4a) |
| Cascading Style Sheet (.css) | JPEG (.jpg) |
| Log File (.log) | H264 (.mp4) |
| Plain Text (.txt) | AVI (.avi) |
| JavaScript (.js) | Windows Media Video (.wmv) |
| Java Source Code (.java) | Flash Video (.flv) |
| XML (.xml) | Shockwave Flash (.swf) |
| HTML (.html) | Waveform Audio (.wav) |
| Active Server Page (.asp) | Windows Media Audio (.wma) |
| Rich Text Language (.rtf) | |
| Thunderbird (.msf/none) | **COMPRESSED-LOSSLESS** |
| | Bi-Tonal Image (.tif/.tiff) |
| **BINARY** | GIF (.gif) |
| Bitmap (.bmp) | Portable Network Graphic (.png) |
| Windows Executable (.exe) | Bzip2 (.bz2) |
| Windows Library (.dll) | Gzip (.gz) |
| Linux Executable (.elf) | ZIP (.zip) |
| | Java Archive (.jar) |
| **BINARY/TEXT** | RPM Package Manger (.rpm) |
| MS Excel 97-2003 (.xls) | PDF (.pdf) |
| MS Word 97-2003 (.doc) | MS Word 2007+ (.docx) |
| MS PowerPoint 97-2003 (.ppt) | MS Excel 2007+ (.xlsx) |
| File System – EXT3/4 | MS PowerPoint 2007+ (.pptx) |
| File System – NTFS | JBIG2 (.jb2) |
| File System – FAT | Outlook PST (.pst) |

experiments were conducted with two different file-to-class assignment methods.

All the samples were 512 B fragments extracted from full files, beginning at file offset 512. The header block was excluded to ensure that the file classifications were not biased by file signatures. File signatures remain the most reliable means to type classify files (when they are present and reliable), but this work focuses on data and file type classification absent traditional, reliable file signatures, file extensions and filesystem data. File fragments used in the experiments were randomly selected from a large corpus of files created by the authors of this chapter for their prior work [9] and augmented with additional types from [7].

*Table 2.* Data and file types in the dataset.

| Data Types | | | | |
|---|---|---|---|---|
| JPG[1] | BZIP2[2] | WMV[2] | HTML[1] | TXT[1,2] |
| GIF[1] | GZIP[1] | MP3[2] | XML[1] | BASE64[2] |
| BMP[1] | DOC[1] | MP4[2] | JSON[2] | BASE85[2] |
| PNG[1] | DOCX[1] | AVI[2] | CSV[1] | URLENCODED[2] |
| TIF[2] | XLS[1] | FLV[2] | CSS[2] | FAT FS DATA[2] |
| PDF[1] | XLSX[1,2] | M4A[2] | LOG[1] | NTFS FS DATA[2] |
| PS[1] | PPT[1] | JAVA[1] | ASP[2] | EXT3 FS DATA[2] |
| ZIP[2] | PPTX[1] | JS[2] | DLL[2] | AES256[2] |
| RTF[2] | PST[2] | JAR[2] | ELF[2] | RANDOM[2] |
| SWF[2] | RPM[2] | JB2[2] | EXE[2] | |
| TBIRD[2] | WAV[2] | WMA[2] | | |

[1] Data Source: GovDocs [15]
[2] Data Source: Other [7]
"FS DATA" = File System Data ($MFT, inode tables, etc.)

The feature set included unigrams, bigrams, entropy, Kolmogorov complexity, mean byte value, Hamming weight, average contiguity between bytes and longest byte streak. Interested readers are referred to [9] for details and equations. The dataset included samples across 52 data and file types. Table 2 lists the data and file types in the dataset.

## 3. Experimentation

The experimental procedure involved several rounds of experiments to empirically test the hypothesized model, following which new models were derived empirically and evaluated comparatively. Each successive experimental round is discussed separately.

### 3.1 Hypothesized Model Testing

The first experimental round involved empirical testing of the hypothesized model. The $k$-means clustering algorithm was applied to 20 randomly-selected samples from each of the 52 data and file types. The value of $k$ was set to six ($k = 6$) to see if the resulting six clusters aligned with the hypothesized file type classes. A simple voting method was used to assign types to classes such that the cluster with the most members of a single type was the winning class for the type. For example, if the cluster distribution for the $n = 20$ JPG samples was $[0, 1, 10, 3, 4, 2]$, then the type JPG was assigned to the third class because the majority of the samples were assigned to the third cluster.

*Table 3.* $k$-means ($k = 6$) clustering solution.

| Class | Data and File Types |
|---|---|
| 0 | B85, AVI, B64, DLL, EXE, MOV, TBIRD |
| 1 | CSS, CSV, DOC, HTML, JAVA, JS, JSON, LOG, PST, RTF, URL, XML |
| 2 | AES, BMP, BZ2, DOCX, ELF, FLV, GIF, GZ, JAR, JB2, JPG, M4A, MP3, MP4, PDF, PNG, PPT, PPTX, RAND, RPM, SWF, WAV, WMV, XLSX, ZIP |
| 3 | EXT3 |
| 4 | NTFS, TIF |
| 5 | B16, FAT, PS, TEXT, XLS |

The type-to-class distribution for the $k$-means ($k = 6$) solution did not align with the hypothesized type-to-class model. This can be seen when comparing Table 3 with Table 1. In particular, the classifier was unable to separately classify lossy compressed files vs. lossless compressed files vs. random and encrypted files, which is where the majority of the deviations from the hypothesized model occurred.

## 3.2 Exploratory Cluster Analysis

The second experimental round applied expectation maximization (EM) clustering to empirically derive the classes from the data, including the number of classes. Unlike the previous round, the number of classes was not set *a priori*. Once again, 20 randomly-selected samples from each type were employed.

*Table 4.* Expectation maximization clustering solution.

| Class | Data and File Types |
|---|---|
| 0 | AES, AVI, BMP, BZ2, DLL, DOCX, ELF, EXE, FLV, GIF, GZ, JAR, JB2, JPG, M4A, MOV, MP3, MP4, PDF, PNG, PPT, PPTX, RAND, RPM, SWF, WAV, WMV, XLSX, ZIP |
| 1 | [EMPTY] |
| 2 | B85, B64, FAT, PST |
| 3 | B16, CSS, CSV, DOC, EXT3, NTFS, HTML, JAVA, JS, JSON, LOG, PS, RTF, TBIRD, TIF, TEXT, URL, XLS, XML |

The expectation maximization results in Table 4 provide intuitive classes; however, Classes 0 and 3 are disproportionately large. Also, Class 1 did not receive any members upon model convergence. Since the

*Table 5.* Sample output supporting the advanced type-to-class assignment method.

| **Data Type** | **Type-to-Cluster Distribution (n=50)** | **Class Assignment** |
|---|---|---|
| EXT | [0, 0, 0, 0, 0, 37, 0, 0, 13, 0] | 6 |
| FAT | [0, 0, 0, 0, 0, 30, 0, 1, 19, 0] | 6 |
| NTFS | [0, 0, 0, 0, 0, 22, 0, 0, 28, 0] | 9 |

primary motivation of hierarchical classification is to significantly reduce the multi-class size at all levels of the hierarchical classification process, large and empty hierarchical classes are undesirable.

In a *post hoc* treatment, Class 0 was further clustered using the $k$-means algorithm ($k = 3$). When selecting 20 instances per data and file type, only the AVI file type separated from the class, leaving two clusters (one with AVI and the other with the remaining 28 types). When selecting 50 instances per data and file type, only ELF and PDF separated out and AVI remained with the remaining 26 types.

Class 3 was clustered similarly with the $k$-means algorithm ($k = 3$). When selecting 20 instances per data and file type, only NTFS and TIF separated out, leaving the other 17 types in a single cluster. When selecting 50 instances per type, EXT3, NTFS and TIF separated out, still leaving the other 16 types in a single cluster. These findings underscore the challenge of statistically distinguishing between (classifying) lossy compressed objects, lossless compressed objects and random and encrypted data.

**Advanced Type-to-Class Assignment Procedure.** In the third experimental round, the number of randomly-selected samples per type was increased from $n = 20$ to $n = 50$ per type, and an advanced method was employed for type-to-class assignment. Instead of using a simple "maximum cluster assignment takes all" voting method, the type-to-cluster distribution matrix was clustered for the final class assignment.

To demonstrate and explain the advanced type-to-class assignment method, a partial type-to-cluster distribution matrix is shown in Table 5; this is taken from a 10-class model that resulted from expectation maximization clustering of 50 samples per type. The simple voting method classified NTFS data as Class 9 and EXT and FAT filesystem data as Class 6, despite the obvious similarity in class membership distributions. This suggests that they might be better classified as a singular class. This is just one example of several other similar situations.

*Table 6.* Expectation maximization clustered matrix approach.

| Class | Data and File Types |
|---|---|
| 0 | CSS, EXT3, FAT, NTFS, TIF, URL, XLS |
| 1 | B85, B16, B64, DOC, HTML, JAR, JAVA, PDF, PPT, PS, RPM |
| 2 | AES, AVI, BMP, BZ2, DLL, DOCX, ELF, EXE, FLV, GIF, GZ, JB2, JPG, M4A, MOV, MP3, MP4, PNG, PPTX, PST, RAND, SWF, WAV, WMV, XLSX, ZIP |
| 3 | CSV, JS, JSON, LOG, RTF, TBIRD, TEXT, XML |

*Table 7.* $k$-means ($k = 6$) clustered matrix approach.

| Class | Data and File Types |
|---|---|
| 0 | B85, AVI, DOC, ELF, EXE, PDF, PPT, PST, WAV |
| 1 | AES, BMP, BZ2, DLL, DOCX, FLV, GIF, GZ, JAR, JB2, JPG, M4A, MOV, MP3, MP4, PNG, PPTX, RAND, RPM, SWF, WMV, XLSX, ZIP |
| 2 | URL |
| 3 | CSV, JSON, LOG, RTF, TBIRD, TEXT |
| 4 | EXT3, FAT, NTFS, TIF |
| 5 | B16, B64, CSS, HTML, JAVA, JS, PS, XLS, XML |

*Table 8.* $k$-means ($k = 4$) clustered matrix approach.

| Class | Data and File Types |
|---|---|
| 0 | B16, CSS, EXT3, FAT, NTFS, PS, TIF, URL, XLS |
| 1 | B85, AVI, DOC, ELF, EXE, PDF, PPT, PST, WAV |
| 2 | B64, CSV, HTML, JAVA, JS, JSON, LOG, RTF, TBIRD, TEXT, XML |
| 3 | AES, BMP, BZ2, DLL, DOCX, FLV, GIF, GZ, JAR, JB2, JPG, M4A, MOV, MP3, MP4, PNG, PPTX, RAND, RPM, SWF, WMV, XLSX, ZIP |

Using the advanced type-to-class membership assignment procedure, the type-to-cluster distribution matrix (comprising 50 samples of each of the 52 types) was clustered. This procedure resulted in the hierarchical models shown in Tables 6 through 9 using various clustering approaches.

Examination of the hierarchical models from a face validity perspective as well as a class balance perspective reveals that the advanced type-to-class assignment method outperforms the simple voting method. Also, meta-classes from across the various experiments can be qualitatively inferred by observing common type-to-cluster classification trends

*Table 9.* $k$-means ($k = 3$) clustered matrix approach.

| Class | Data and File Types |
|---|---|
| 0 | B64, CSV, HTML, JAVA, JS, JSON, LOG, RTF, TBIRD, TEXT, XML |
| 1 | B16, CSS, ELF, EXT3, FAT, NTFS, PDF, PS, TIF, URL, XLS |
| 2 | B85, AES, AVI, BMP, BZ2, DLL, DOC, DOCX, EXE, FLV, GIF, GZ, JAR, JB2, JPG, M4A, MOV, MP3, MP4, PNG, PPT, PPTX, PST, RAND, RPM, SWF, WAV, WMV, XLSX, ZIP |

*Table 10.* Meta-classes from hierarchical models.

| Meta-Class | Data and File Types |
|---|---|
| Compressed | PPTX, RAND, AES, JB2, ZIP, XLSX, JPG, DLL, BMP, GZ, SWF, DOCX, GIF, MP4, MOV, MP3, PNG, M4A, FLV, WMV, BZ2 |
| Text | TBIRD, XML, RTF, LOG, TEXT, JS, JSON, CSV |
| Binary-Text | NTFS, EXT3, URL, FAT, TIF, XLS, CSS |

(see Table 10). However, the 16 types fail to consistently cluster into a stable hierarchy: B64, HTML, JAVA, LOG, B16, TIF, URL, XLS, B85, AVI, DOC, EXE, GIF, GZ, JB2 and WAV. Hence, while the meta-classes shown might be useful for a two-level hierarchical model limited to its constituent 36 file and data types, further exploratory analysis is needed to derive a more appropriate hierarchical model for all 52 file and data types considered in this work.

## 3.3 Identifying the Winning Model

The purpose of the final experimental round was to run repeated experiments for more robust results, from which a winning model would be selected. Three repeated experiments were conducted and quantitative measures of model quality were evaluated comparatively. Expectation maximization clustering was performed on the same input three times. Then, the simple voting method and the advanced assignment method described previously were applied to each set of expectation maximization clustered outputs. The prediction accuracy, precision and recall were then computed and evaluated.

*Table 11.* Repeated experiments (Round 1; simple voting method).

| Class | Types Clustered into Class |
|---|---|
| 0 | HTML, JAVA, JS |
| 1 | JSON |
| 2 | B85, B16, B64, TBIRD, URL |
| 3 | TIF |
| 4 | EXT3, FAT |
| 5 | CSV, LOG, PS, RTF, TEXT, XML |
| 6 | CSS, DOC, ELF, NTFS, XLS |
| 7 | AVI, BMP, DLL, EXE, JAR, MOV, WAV |
| 8 | AES, BZ2, DOCX, FLV, GIF, GZ, JB2, JPG, M4A, MP3, MP4, PDF, PNG, PPT, PPTX, PST, RAND, RPM, SWF, WMV, XLSX, ZIP |

*Table 12.* Repeated experiments (Round 1; advanced assignment method).

| Class | Types Clustered into Class |
|---|---|
| 0 | CSS, ELF, EXT3, FAT, NTFS, XLS |
| 1 | AES, BZ2, FLV, GIF, GZ, JB2, JPG, M4A, MP3, MP4, PDF, PNG, PPT, PPTX, RAND, RPM, SWF, WMV, XLSX, ZIP |
| 2 | B85, B16, B64, TBIRD, URL |
| 3 | CSV, HTML, JAVA, JS, JSON, LOG, PS, RTF, TEXT, XML |
| 4 | AVI, BMP, DLL, DOC, DOCX, EXE, JAR, MOV, PST, TIF, WAV |

*Table 13.* Repeated experiments (Round 2; simple voting method).

| Class | Types Clustered into Class |
|---|---|
| 0 | CSS, NTFS, XLS |
| 1 | B85, ELF, MOV, WAV |
| 2 | AES, AVI, BMP, BZ2, DLL, DOC, DOCX, EXE, FLV, GIF, GZ, JAR, JB2, JPG, M4A, MP3, MP4, PDF, PNG, PPT, PPTX, PST, RAND, RPM, SWF, WMV, XLSX, ZIP |
| 3 | CSV, JSON, LOG |
| 4 | EXT3, FAT |
| 5 | URL |
| 6 | TIF |
| 7 | B16, B64, HTML, JAVA, JS, PS, RTF, TBIRD, TEXT, XML |

Tables 11 through 16 present the results of the repeated experiments. The results of each experiment were evaluated considering face validity

*Table 14.* Repeated experiments (Round 2; advanced assignment method).

| Class | Types Clustered into Class |
|---|---|
| 0 | B16, B64, CSV, HTML, JAVA, JS, JSON, LOG, PS, RTF, TBIRD, TEXT, XML |
| 1 | B85, AVI, BMP, CSS, DLL, DOC, ELF, EXE, EXT3, FAT, NTFS, MOV, PDF, TIF, URL, WAV, XLS |
| 2 | AES, BZ2, DOCX, FLV, GIF, GZ, JAR, JB2, JPG, M4A, MP3, MP4, PNG, PPT, PPTX, PST, RAND, RPM, SWF, WMV, XLSX, ZIP |

*Table 15.* Repeated experiments (Round 3; simple voting method).

| Class | Types Clustered into Class |
|---|---|
| 0 | CSS, ELF, NTFS, XLS |
| 1 | B85, B16, B64, CSV, HTML, JAVA, JS, JSON, LOG, PS, RTF, TBIRD, TEXT, URL, XML |
| 2 | AES, AVI, BMP, BZ2, DLL, DOC, DOCX, EXE, FLV, GIF, GZ, JAR, JB2, JPG, M4A, MOV, MP3, MP4, PDF, PNG, PPT, PPTX, PST, RAND, RPM, SWF, WAV, WMV, XLSX, ZIP |
| 3 | EXT3, FAT, TIF |

*Table 16.* Repeated experiments (Round 3; advanced assignment method).

| Class | Types Clustered into Class |
|---|---|
| 0 | B85, B16, B64, CSV, HTML, JAVA, JS, JSON, LOG, PS, RTF, TBIRD, TEXT, URL, XML |
| 1 | BMP, CSS, DLL, DOC, ELF, EXE, EXT3, FAT, NTFS, PDF, TIF, XLS |
| 2 | AES, AVI, BZ2, DOCX, FLV, GIF, GZ, JAR, JB2, JPG, M4A, MOV, MP3, MP4, PNG, PPT, PPTX, PST, RAND, RPM, SWF, WAV, WMV, XLSX, ZIP |

and quantitative measures (overall model classification accuracy as well as type-level precision and recall). This validated the interim conclusion that the advanced type-to-class distribution matrix clustering approach outperforms the simple voting method for type-to-class assignment.

Upon comparing the quantitative measures across the three experiments for the type-to-class distribution matrix clustering method, it

*Table 17.* Winning hierarchical model.

| Class: Title | File and Data Types Assigned to Class |
|---|---|
| 0: Binary-Text | CSS, ELF, EXT3, FAT, NTFS, XLS |
| 1: Compressed-Random | AES, BZ2, FLV, GIF, GZ, JB2, JPG, M4A, MP3, MP4, PDF, PNG, PPT, PPTX, RAND, RPM, SWF, WMV, XLSX, ZIP |
| 2: Encoded Text | B85, B16, B64, TBIRD, URL |
| 3: Unencoded Text | CSV, HTML, JAVA, JS, JSON, LOG, PS, RTF, TEXT, XML |
| 4: Binary-Compressed | AVI, BMP, DLL, DOC, DOCX, EXE, JAR, MOV, PST, TIF, WAV |

can be concluded that the hierarchy shown in Table 12 is the winning hierarchy. Table 17 redisplays the winning hierarchy with rough class descriptive titles. Note that the titles should be used with caution because they do not describe all the constituent file types in an ideal manner.

*Table 18.* Performance measures – Class-level classification.

| Model | Train Time | Predict Time | Accuracy (No. Obs.) | C |
|---|---|---|---|---|
| Hypothesized | 3m30.570s | 0m08.705s | 76.51% (62,426) | 64 |
| Table 12 | 2m22.398s | 0m12.317s | 88.88% (62,322) | 512 |
| Table 16 | 1m13.641s | 0m11.018s | 94.40% (62,081) | 1,024 |

Table 18 presents the comparative sample-to-class classification predication accuracy and train/test run times for the hypothesized model and the two top-performing empirically-derived models, the Table 12 model and the Table 16 model. Tables 19 through 21 provide similar data for the two models; however, the values pertain to type-level, within-class classification.

## 4. Winning Model Discussion

The hierarchical model shown in Table 16 exhibits superior performance when prediction accuracy is considered only at the class level. The class-level classification accuracy of the theoretical model is 76.51%. The classification accuracy for the model shown in Table 12 (i.e., the model selected as the winning hierarchy) is 88.88%. The accuracy improves to 94.40% for the model shown in Table 16.

*Table 19.* Performance measures – Type-level classification (Hypothesized model).

| Class | Train Time | Predict Time | Accuracy (No. Obs.) | C |
|---|---|---|---|---|
| 1 | 0m34.001s | 0m2.471s | 95.21% (19,638) | 512 |
| 2 | 0m07.526s | 0m0.047s | 0.00% (9) | 1,024 |
| 3 | 0m13.168s | 0m0.414s | 65.54% (5,613) | 64 |
| 4 | 0m48.827s | 0m1.853s | 75.60% (5,291) | 32 |
| 5 | 0m08.798s | 0m0.732s | 85.85% (3,902) | 128 |
| 6 | 1m25.891s | 0m5.594s | 23.17% (27,973) | 32 |

*Table 20.* Performance measures – Type-level classification (Table 12 model).

| Model | Train Time | Predict Time | Accuracy (No. Obs.) | C |
|---|---|---|---|---|
| 0 | 0m05.469s | 0m00.661s | 71.10% (7,453) | 32 |
| 1 | 2m12.123s | 0m10.278s | 24.81% (28,081) | 512 |
| 2 | 0m05.003s | 0m00.982s | 97.96% (5,918) | 512 |
| 3 | 0m11.236s | 0m00.752s | 90.88% (12,609) | 512 |
| 4 | 1m13.674s | 0m01.882s | 85.64% (8,261) | 512 |

*Table 21.* Performance measures – Type-level classification (Table 16 model).

| Model | Train Time | Predict Time | Accuracy (No. Obs.) | C |
|---|---|---|---|---|
| 0 | 0m31.645s | 0m02.387s | 94.25% (18,509) | 1,024 |
| 1 | 1m18.770s | 0m01.267s | 77.71% (11,855) | 64 |
| 2 | 4m29.957s | 0m11.186s | 35.59% (31,717) | 32 |

While the class-level classification accuracy of the model selected as the winning model (Table 12) is not maximal (it is less than that of the model shown in Table 16), its average within-class, type-level classification accuracy values exceed those of the other candidate models. The average type-level classification accuracy values are: theoretical model – 57.56%; winning model (Table 12) – 74.08%; and the model shown in Table 16 – 69.18%. These differing results are presented because some use cases may prefer to sacrifice type-level classification accuracy for class-level classification accuracy, while others may prefer the converse.

In selecting the model reflected by Table 12 as the winning model, the fact that 25 high-entropy file types classified in the most poorly-performing class in the Table 16 model was considered. Since this is

one of only three classes in the model, having such a large, poorly-performing class is problematic. In contrast, the Table 12 model contains five classes and the most poorly-performing class (also characterized by high-entropy file types) only contains 20 file types. This is important because a hierarchical classification system that contains a singular, large, poorly-performing, high-entropy class among very few classes is only effective at the class level; its utility for type-level classification would be very limited. Therefore, it is important to minimize the number of file types in such a class. Hence, the Table 12 hierarchy is preferred to the Table 16 hierarchy – it contains more classes and fewer constituent file types in the characteristically poorly-performing, high-entropy class.

Hierarchical models with more classes are also favored because they provide greater analytical granularity at the class level than models with more types in fewer classes. For example, a possible application of class-level classification is network-based data triage and prioritization of limited deep packet inspection resources. Inbound packet payloads could be classified and marked for deep packet inspection consideration if the packet payload is classified at the class level in a particular category. Accordingly, the Table 12 model is favored over the Table 16 model.

Specific file types classified in the poorly-performing, high-entropy class for the Table 12 model were also examined. The only file type that may be better served by being in a different class is PDF. In contrast, in the Table 16 model, six file types may be better served by being in a different class: AVI, WAV, DOCX, JAR, MOV and PST.

Finally, the file-level recall and precision were comparatively evaluated for the experiments involving the top-two candidate hierarchical models – the Table 12 and Table 16 models (see Tables 22 and 23). The results are not compelling for one model over the other, but it is clear that the Table 12 model slightly outperforms the Table 16 model on average. Considering both recall and precision as equally important, the Table 12 model yields better results than the Table 16 model in a slight majority of cases (where there is a difference at all).

## 5. Limitations and Future Research

Given the alternative findings on the impact of feature vector selection [9], it is unknown if feature vector selection will impact hierarchical modeling. Future research should explore the impact of varying feature vectors, either to reliably converge on a winning hierarchy or to determine if different hierarchical models require different input feature vectors or whether different features are better predictors for different classes. Future research should also explore the impact of reduced di-

*Table 22.* Comparative type-level recall and precision.

| Type | Table 12 Recall | Table 12 Precision | Table 16 Recall | Table 16 Precision |
|---|---|---|---|---|
| TXT | 0.979 | 0.793 | 0.968 | 0.830 |
| CSV | 0.988 | 0.979 | 0.986 | 0.962 |
| LOG | 0.985 | 0.933 | 0.990 | 0.933 |
| HTML | 0.897 | 0.749 | 0.901 | 0.874 |
| XML | 0.943 | 0.970 | 0.945 | 0.974 |
| JSON | 0.984 | 0.993 | 0.974 | 0.998 |
| JS | 0.935 | 0.904 | 0.940 | 0.908 |
| JAVA | 0.961 | 0.890 | 0.972 | 0.918 |
| CSS | 1.000 | 0.979 | 1.000 | 0.978 |
| B64 | 1.000 | 0.983 | 1.000 | 0.987 |
| B85 | 1.000 | 0.975 | 1.000 | 0.961 |
| B16 | 1.000 | 0.975 | 0.999 | 0.976 |
| URL | 1.000 | 0.999 | 1.000 | 0.998 |
| PS | 0.986 | 0.944 | 0.983 | 0.942 |
| RTF | 0.998 | 0.979 | 0.974 | 0.978 |
| TBIRD | 1.000 | 0.950 | 0.976 | 0.924 |
| PST | 0.934 | 0.971 | 0.456 | 0.473 |
| PNG | 0.007 | 0.105 | 0.001 | 0.500 |
| GIF | 0.876 | 0.326 | 0.873 | 0.338 |
| TIF | 0.964 | 0.837 | 0.982 | 0.826 |
| JB2 | 0.318 | 0.134 | 0.006 | 0.167 |
| GZ | 0.047 | 0.206 | 0.001 | 0.048 |
| ZIP | 0.003 | 0.500 | 0.001 | 1.000 |
| JAR | 0.963 | 0.736 | 0.380 | 0.385 |
| RPM | 0.527 | 0.148 | 0.330 | 0.188 |
| BZ2 | 0.304 | 0.236 | 0.751 | 0.196 |
| PDF | 0.438 | 0.333 | 0.995 | 0.977 |
| DOCX | 0.975 | 0.757 | 0.560 | 0.613 |
| XLSX | 0.499 | 0.639 | 0.525 | 0.474 |
| PPTX | 0.000 | 0.000 | 0.003 | 0.250 |
| JPG | 0.507 | 0.242 | 0.649 | 0.236 |
| MP3 | 0.910 | 0.324 | 0.679 | 0.438 |
| M4A | 0.544 | 0.182 | 0.084 | 0.221 |
| MP4 | 0.403 | 0.387 | 0.598 | 0.240 |
| AVI | 0.931 | 0.863 | 0.828 | 0.614 |
| WMV | 0.020 | 0.571 | 0.614 | 0.199 |
| FLV | 0.479 | 0.223 | 0.313 | 0.406 |

mension feature vectors [2, 3] on hierarchical classification modeling, considering the trade-off between computational costs and classification accuracy. Finally, research should consider replicating the findings pre-

*Table 23.* Comparative type-level recall and precision (cont'd.).

| Type | Table 12 Recall | Table 12 Precision | Table 16 Recall | Table 16 Precision |
|---|---|---|---|---|
| WAV | 0.981 | 0.896 | 0.947 | 0.780 |
| MOV | 0.997 | 0.963 | 0.971 | 0.961 |
| DOC | 0.774 | 0.599 | 0.747 | 0.662 |
| XLS | 0.943 | 0.893 | 0.875 | 0.962 |
| PPT | 0.001 | 0.023 | 0.003 | 0.094 |
| FAT | 0.308 | 0.773 | 0.285 | 0.819 |
| NTFS | 0.735 | 0.854 | 0.713 | 0.948 |
| EXT3 | 0.975 | 0.449 | 0.958 | 0.458 |
| EXE | 0.766 | 0.764 | 0.703 | 0.678 |
| DLL | 0.857 | 0.806 | 0.827 | 0.842 |
| ELF | 0.927 | 0.723 | 0.877 | 0.856 |
| BMP | 0.882 | 0.929 | 0.870 | 0.904 |
| AES | 0.038 | 0.109 | 0.000 | 0.000 |
| RAND | 0.000 | 0.000 | 0.002 | 0.063 |

sented in this work with alternate samples and/or more files per type during experimentation. This work would likely have to be facilitated by distributed computing given the computational costs associated with high-dimension vectors (65,500+ dimensions) and a large number of (2,600) observations (50 types $\times$ 52 files per type).

## 6. Conclusions

This research demonstrates the utility of a multi-step approach for improving data and file type classification performance. The approach provides a means to classify inputs at the class level, which is faster than type-level classification (when the number of total types is held constant), and class-level classification may be adequate in some applications. It can also improve type-level classification by simply reducing the multi-class size of the sub-classification problems (once again, when the number of total types is held constant). In other applications, class-level classification may serve as a useful triage step to direct limited deep packet inspection resources or when applying specialized classification techniques, such as those advocated in [24]. In fact, one might contend that the optimal classification approach is a hybrid approach (combining statistical and specialized approaches) selected during the multi-level class-to-type hierarchical classification process. For example, a quick $n$-gram-based statistical classification for detecting an input

as compressed (class-level) followed by specialized techniques for distinguishing between a PDF fragment from a PPTX fragment (type-level) may be the optimal approach, despite the body of research that takes a "one-size-fits-all," non-hierarchical, type-level classification approach.

This research has empirically tested and invalidated the hypothesized six-class, two-level hierarchy and has used exploratory analysis to empirically derive a winning two-level, five-class hierarchical model. A total of 52 file and data types were considered and file header blocks were excluded from the procedure to ensure that the file signatures did not bias the classification results. File header blocks were also excluded because the research focused on classifying file fragments absent reliable file signatures, file extensions or other filesystem data. The experimental results demonstrate that a two-level (class and type) classification hierarchical model is both feasible and advantageous. Moreover, the approach leads to very good class-level classification performance, improved classification performance for data and file types not exhibiting high entropy (e.g., compressed and encrypted data) and reasonably equivalent classification performance for high-entropy data and file types.

Note that the views expressed in this chapter do not necessarily reflect the official policies of the Naval Postgraduate School nor does the mention of trade names, commercial practices or organizations imply an endorsement by the U.S. Government.

## Acknowledgement

This research was supported by the Naval Postgraduate School Assistance Grant/Agreement No. N00244-13-1-0027 awarded by the NAVSUP Fleet Logistics Center San Diego (NAVSUP FLC San Diego).

## References

[1] I. Ahmed, K. Lhee, H. Shin and M. Hong, On improving the accuracy and performance of content-based file type identification, *Proceedings of the Fourteenth Australasian Conference on Information Security and Privacy*, pp. 44–59, 2009.

[2] I. Ahmed, K. Lhee, H. Shin and M. Hong, Fast file type identification, *Proceedings of the ACM Symposium on Applied Computing*, pp. 1601–1602, 2010.

[3] I. Ahmed, K. Lhee, H. Shin and M. Hong, Fast content-based file type identification, in *Advances in Digital Forensics VII*, G. Peterson and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 65–75, 2011.

[4] M. Amirani, M. Toorani and A. Beheshti, A new approach to content-based file type detection, *Proceedings of the IEEE Symposium on Computers and Communications*, pp. 1103–1108, 2008.

[5] M. Amirani, M. Toorani and S. Mihandoost, Feature-based type identification of file fragments, *Security and Communication Networks*, vol. 6(1), pp. 115–128, 2013.

[6] S. Axelsson, Using normalized compression distance for classifying file fragments, *Proceedings of the International Conference on Availability, Reliability and Security*, pp. 641–646, 2010.

[7] N. Beebe, UTSA Filetypes 1 Data Set, Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, Texas (`digitalcorpora.org/corp/files/filetypes1`), 2016.

[8] N. Beebe and L. Liu, Ranking algorithms for digital forensic string search hits, *Digital Investigation*, vol. 11(S2), pp. S124–S132, 2014.

[9] N. Beebe, L. Maddox, L. Liu and M. Sun, Sceadan: Using concatenated $n$-gram vectors for improved file and data type classification, *IEEE Transactions on Information Forensics and Security*, vol. 8(9), pp. 1519–1530, 2013.

[10] W. Calhoun and D. Coles, Predicting the types of file fragments, *Digital Investigation*, vol. 5(S), pp. S14–S20, 2008.

[11] D. Cao, J. Luo, M. Yin and H. Yang, Feature-selection-based file type identification algorithm, *Proceedings of the IEEE International Conference on Intelligent Computing and Intelligent Systems*, vol. 3, pp. 58–62, 2010.

[12] G. Conti, S. Bratus, A. Shubina, B. Sangster, R. Ragsdale, M. Supan, A. Lichtenberg and R. Perez-Alemany, Automated mapping of large binary objects using primitive fragment type classification, *Digital Investigation*, vol. 7(S), pp. S3–S12, 2010.

[13] R. Erbacher and J. Mulholland, Identification and localization of data types within large-scale filesystems, *Proceedings of the Second International Workshop on Systematic Approaches to Digital Forensic Engineering*, pp. 55–70, 2007.

[14] S. Fitzgerald, G. Mathews, C. Morris and O. Zhulyn, Using NLP techniques for file fragment classification, *Digital Investigation*, vol. 9(S), pp. S44–S49, 2012.

[15] S. Garfinkel, P. Farrell, V. Roussev and G. Dinolt, Bringing science to digital forensics with standardized forensic corpora, *Digital Investigation*, vol. 6(S), pp. S2–S11, 2009.

[16] S. Gopal, Y. Yang, K. Salomatin and J. Carbonell, Statistical learning for file type identification, *Proceedings of the Tenth International Conference on Machine Learning and Applications*, vol. 1, pp. 68–73, 2011.

[17] G. Hall and W. Davis, Sliding Window Measurement for File Type Identification, Technical Report, Department of Computer Science, Texas State University-San Marcos, San Marcos, Texas, 2006.

[18] M. Karresand and N. Shahmehri, File type identification of data fragments by their binary structure, *Proceedings of the IEEE Information Assurance Workshop*, pp. 140–147, 2006.

[19] M. Karresand and N. Shahmehri, Oscar – File type identification of binary data in disk clusters and RAM pages, *Proceedings of the Twenty-First IFIP TC-11 International Conference on Information Security*, pp. 413–424, 2006.

[20] Q. Li, A. Ong, P. Suganthan and V. Thing, A novel support vector machine approach to high-entropy data fragment classification, *Proceedings of the South African Information Security Multi-Conference*, 2011.

[21] W. Li, K. Wang, S. Stolfo and B. Herzog, Fileprints: Identifying file types by $n$-gram analysis, *Proceedings of the Sixth Annual IEEE SMC Information Assurance Workshop*, pp. 64–71, 2005.

[22] M. McDaniel and M. Heydari, Content-based file type detection algorithms, *Proceedings of the Thirty-Sixth Annual Hawaii International Conference on System Sciences*, 2003.

[23] S. Moody and R. Erbacher, SADI – Statistical analysis for data type identification, *Proceedings of the Third International Workshop on Systematic Approaches to Digital Forensic Engineering*, pp. 41–54, 2008.

[24] V. Roussev and S. Garfinkel, File fragment classification – The case for specialized approaches, *Proceedings of the Fourth IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering*, pp. 3–14, 2009.

[25] C. Veenman, Statistical disk cluster classification for file carving, *Proceedings of the Third International Symposium on Information Assurance and Security*, pp. 393–398, 2007.

[26] L. Zhang and G. White, An approach to detect executable content for anomaly-based network intrusion detection, *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing*, 2007.