Faster classification using compression analytics

Christina Ting*, Nicholas Johnson†, Uzoma Onunkwo‡, and J. Derek Tucker§
Sandia National Laboratories
Albuquerque, NM 87123
Email: *clting@sandia.gov, †nicjohn@sandia.gov, ‡uonunkw@sandia.gov, §jdtuck@sandia.gov

***Abstract*—Compression analytics have gained recent interest for application in malware classification and digital forensics. This interest is due to the fact that compression analytics rely on measured similarity between byte sequences in datasets without requiring prior feature extraction; in other words, these methods are *featureless*. Being featureless makes compression analytics particularly appealing for computer security applications, where good static features are either unknown or easy to circumvent by adversaries. However, previous classification methods based on compression analytics relied on algorithms that scaled with the size of each labeled class and the number of classes. In this work, we introduce an approach that, in addition to being featureless, can perform fast and accurate inference that is independent of the size of each labeled class. Our method is based on calculating a representative sample, the Fréchet mean, for each labeled class and using it at inference time. We introduce a greedy algorithm for calculating the Fréchet mean and evaluate its utility for classification across a variety of computer security applications, including authorship attribution of source code, file fragment type detection, and malware classification.**

I. Introduction

Machine learning continues to enjoy success in an ever-growing range of disciplines, from bioinformatics to computer security. Traditionally, machine learning requires the extraction of data features, which are used to create decision boundaries for classification of data. As an example, prototype methods such as k-nearest-neighbors (k-NN) are a class of machine learning methods that label unknown data items according to proximity to known (labeled) items. These methods can be very effective and are often among the best performers on real data problems [1]. However, the performance of these methods depends critically on the metric used to define proximity between items; a standard metric choice is the Euclidean distance over the data features [2].

In computer security applications such as malware classification and digital forensics, selecting adequate features can be daunting and ambiguous even when excellent domain knowledge of features exists. For instance, in malware classification, adversaries are highly motivated to change their approach over time so as to reduce the chances of detection [3]–[6]; in digital forensics [7]–[9], features such as byte frequency distribution often change as file formats evolve. If the relationships between data features change, a previously appropriate metric may no longer yield high quality decision boundaries. Thus, extracting a useful feature set is paramount but can be particularly challenging in computer security applications.

Alternatively, rather than defining metrics over data features, *featureless* metrics operate directly on the data items themselves. In particular, the *normalized information distance* (NID) is a distance metric[1] for any two data items because it measures the minimum quantity of information sufficient to translate between them [10]. However, the NID is not computable [11]. Therefore, practical applications of the NID are based on the *normalized compression distance* (NCD), which relies on compression algorithms. The NCD between two items, a and b, is defined by [12]

$$\mathrm{NCD}(a,b) = \frac{|C(ab)| - \min\{|C(a)|, |C(b)|\}}{\max\{|C(a)|, |C(b)|\}}, \qquad (1)$$

where $|C(a)|$ denotes the compressed size of a after applying a compression algorithm C, and ab denotes the concatenation of item a followed by item b. If C satisfies the properties of a normal compressor, then the NCD satisfies the properties of a metric (see [13], which also includes the definition of a normal compressor).

The generality of real world compression algorithms in Eq. (1) has led to the success of the NCD in a variety of applications. Examples include authorship attribution [14], image registration [15], evolutionary history inference [12], and cybersecurity applications [16]–[21]. However, even with the previous successes of NCD, there are several shortcomings that should be mentioned. In practice, C in Eq. (1) rarely satisfies the conditions for a normal compressor [13]. For this reason, the NCD is not a metric. Furthermore, the overhead of the full compression algorithm in Eq. (1) renders the NCD impractical for application on large datasets.

Fortunately, recent results have shown that the computational complexity of approximating the NID can be significantly reduced by operating directly on the underlying dictionaries constructed by the compression algorithms [22]–[26]. In [25], Raff and Nicholas introduced the *Lempel-Ziv Jaccard Distance* (LZJD) as an alternative to approximating the NID that has several advantages over the NCD. Specifically, the LZJD is a metric [27]; is orders of magnitude faster than the NCD because it only generates a dictionary instead of compressing the input; and demonstrates comparable and even superior performance to NCD for certain applications.

However, to the best of our knowledge, current prototype implementations using the LZJD for making predictions still require computing the distances between the new unlabeled item and *all* labeled items in the training set, which is highly

[1] A metric must satisfy the identity, symmetry, and triangle inequality properties.

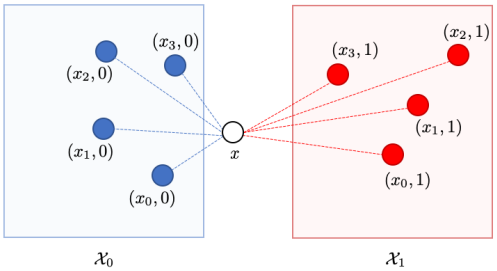

Fig. 1. A prototype method based on distances to *observed* data items. Schematic of a binary classification problem for unknown item x.

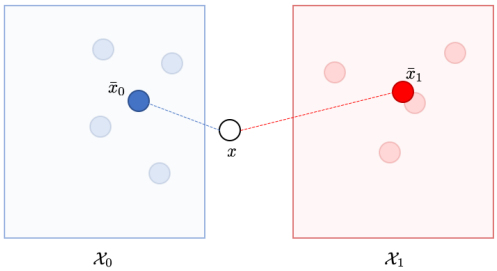

Fig. 2. A prototype method based on distances to *representative* data items. Schematic of a binary classification problem for unknown item x.

undesirable for most applications. In the standard machine learning train and deploy setting, one trains a model on available labeled data then deploys it to makes predictions on unlabeled data that vastly outnumber the labeled data. The train-deploy paradigm also means that more powerful computers can be employed during training than inference so that the requirements at each stage do not have to be equal. Furthermore, with the rise of the internet of things, progressively more devices are attempting to perform inference at the edge (performing predictions where data is collected rather than sending data back to a central computer to perform predictions). In these scenarios, it is worthwhile to trade slower training for faster inference.

In this work, we introduce a prototype implementation using the LZJD that can perform accurate inference while ensuring a constant-time dependence with respect to the size of each labeled class. The method is based on computing the representative *Fréchet mean* of each class in the training set. Inference then requires only a single distance calculation against the Fréchet mean for each class. Providing the ability to make predictions more quickly and simply, with minimal compromises on performance, expands the scale of problems to be addressed. Our main contributions are as follows:

1) We develop a greedy algorithm for approximating the Fréchet mean in the LZJD metric space.
2) We integrate the Fréchet mean as the representative item in prototype methods for efficient inference using the LZJD.
3) We demonstrate, on a variety of datasets, comparable accuracies and superior inference times with the Fréchet mean, compared with predictions requiring the full training set, e.g., k-NN.

In the remainder of this document, we formally discuss prototype methods and describe the Fréchet mean along with an efficient method for approximating it in Section II. In Section III, we describe the variety of datasets that we used in our experiments to evaluate the performance of our approximated Fréchet mean in prototype methods. The results from our experiments are presented in Section IV. We conclude our work and propose areas for future work in Section V.

II. Methods

Consider a training set of n labeled data items belonging to m classes, $m \ll n$. The training set is represented by the pairs $\mathcal{X} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where each y_i is a class label that can take values $j \in \{0, \ldots, m-1\}$. Let $\mathcal{X}_j \subset \mathcal{X}$ be the subset of $\mathcal{X}$ whose items x_i have corresponding class label $y_i = j$. Prototype methods represent the training set as points in feature space and classify a new data item x by considering the classes of the nearby items, where nearby is defined in terms of a distance metric d (we use the Jaccard distance throughout the paper). The nearby items considered are termed prototypes and traditionally consist of the n observed data items.

Examples of prototype methods using observed data items include k-nearest neighbors (k-NN) [28] and average distance [29]. In the average distance classifier, predictions are made by assigning the unknown item, x, to the class that produces, on average, the minimum distance between its items $\mathcal{X}_j$ and x, that is,

$$\hat{y} = \text{argmin}_{j=0,\ldots,m-1} \left[\frac{1}{|\mathcal{X}_j|} \sum_{x_i \in \mathcal{X}_j} d(x_i, x) \right], \qquad (2)$$

where d is the distance metric and $\hat{y}$ is the predicted label of item x. Figure 1 shows a schematic of a prototype method based on distances (dashed lines) to observed data items.

As described in Section I, there are two challenges with prototype methods based on distances to observed data items. First, choosing an appropriate d for applications with an unknown or changing feature space is not straightforward. Section II-A describes the LZJD, an approximation to the NID that bypasses the need for feature selection and has shown success on previous classification tasks [25]. Second, prototype methods based on distances to all observed data items, $\mathcal{X}_j$, $j = 0, ..., m-1$, can be quite expensive in the computation time during inference and memory cost overall. Prototype methods based on distances to the representative item, $\bar{x}_j$, for observed $\mathcal{X}_j$ have a significant advantage. Specifically, instead of n comparisons, one only needs to make m comparisons:

$$\hat{y} = \text{argmin}_{j=0,\ldots,m-1} d(\bar{x}_j, x), \qquad (3)$$

where $m \ll n$. Section II-B introduces the Fréchet mean in the Jaccard distance metric space as a proposed representative

item. Figure 2 shows a schematic of a prototype method based on distances (dashed lines) to representative data items. In this work, we only consider the notion of a single representative item per class, but it is certainly possible to consider other fixed number of representative items per class that is still much less than the average number of observed items per class.

A. Lempel-Ziv Jaccard Distance (LZJD)

Dictionary-based methods for approximating the NID are inspired by the Lempel-Ziv (LZ) algorithm [30], [31] for creating a dictionary of previously seen subsequences. These methods [22]–[26] recognize that it is not necessary to obtain the actual compressed output of the data items. Therefore it is advantageous to skip the technical details required for effective compression and to focus on the dictionary creation.

The Lempel-Ziv Jaccard Distance (LZJD) is a measure of distance defined over the aforementioned dictionaries. The dictionaries are obtained from transformations using a method that relies on a simplified version of the LZ77 [30] algorithm (c.f. Algorithm 1 in [25]) and consistently converts a data item a into a dictionary of sub-sequences $L(a) = A$. In this document, we will refer to A and B as the LZSets of the data items a and b respectively. The LZJD between two data items a and b is then defined by the Jaccard distance on their LZSets, that is,

$$d(A,B) = \begin{cases} 1 - \frac{|A \cap B|}{|A \cup B|} & \text{if } A \cup B \neq \varnothing, \\ 1 & \text{otherwise.} \end{cases} \tag{4}$$

It is known that sets along with the Jaccard distance function form a metric space [27], which is a requirement for the Fréchet mean [32].

For large datasets, computing the pairwise LZJD may be expensive and fast approximations based on the LZJD over hashed LZSets can be used [25], [33]. In this work, we focus on the exact definition of the LZJD over LZSets.

B. The Fréchet mean in the Jaccard distance metric space

Here, we define the Fréchet mean and introduce a greedy algorithm for approximating it. Consider the cost function, $J(M)$, defined in terms of squared LZJD over a list of observed LZSets $X_1, \ldots, X_n$, that is,

$$\begin{aligned} J(M) &= \sum_{i=1}^{n} d^2(M, X_i) \\ &= \sum_{i=1}^{n} \left(1 - \frac{|M \cap X_i|}{|M \cup X_i|}\right)^2 \end{aligned} \tag{5}$$

where M is a set in the domain of the observed LZSets. Then, the Fréchet mean μ of these observed LZSets $X_1, \ldots, X_n$, is the set that minimizes the cost function, $J(M)$:

$$\begin{aligned} \mu &= \arg\min_{M \subset \mathcal{S}} J(M) \\ &= \arg\min_{M \subset \mathcal{S}} \sum_{i=1}^{n} \left(1 - \frac{|M \cap X_i|}{|M \cup X_i|}\right)^2 \end{aligned} \tag{6}$$

where $\mathcal{S}$ is the universal set over the domain of the observed LZSets, that is, $\mathcal{S}$ contains not only the observed sets, but also the unobserved sets in the problem domain.[2]

To constrain the search space, we note that μ is a superset of the intersection ($I = \cap_i X_i$) and a subset of the union ($U = \cup_i X_i$) of the observed LZSets. That is, $I \subseteq \mu \subseteq U$. To show that μ is a superset of I, note that *adding* an element ($e \notin M$) to the candidate set from I strictly decreases the cost function we aim to minimize in Eq. (6) since

$$1 - \frac{|M \cap X_i| + 1}{|M \cup X_i|} < 1 - \frac{|M \cap X_i|}{|M \cup X_i|}.$$

Additionally, *removing* an element from the candidate set that is contained in I strictly increases our cost since

$$1 - \frac{|M \cap X_i| - 1}{|M \cup X_i|} > 1 - \frac{|M \cap X_i|}{|M \cup X_i|}.$$

Similar reasoning can be used to show that μ is a subset of U.

Even so, $|U|$ is, in general, large and a brute-force search is inefficient with a runtime complexity of $O(2^{|U|})$. Therefore, we introduce a greedy algorithm for approximating the Fréchet mean that further reduces the search space to $O(|U|)$. Our greedy algorithm is as follows:

1) Identify *elements* in U, the union of observed LZSets: $\{e_1, \ldots, e_{|U|}\} = \cup_i X_i$.
2) Sort the elements in order of their increasing sum of squared Jaccard distance to observed LZSets, defined by
$$r(e_q) = \sum_{i=1}^{n} d(\{e_q\}, X_i)^2, \tag{7}$$
so that $r(e'_1) \leq r(e'_2) \ldots \leq r(e'_{|U|})$.
3) Define $M^{(1)} = \{e'_1\}$. Incrementally add the next sorted element to the set
$$M^{(q+1)} = M^{(q)} \cup \{e'_{q+1}\}$$
to obtain
$$M^{(1)}, \ldots, M^{(|U|)} = \{e'_1\}, \ldots, \{e'_1, e'_2, \ldots, e'_{|U|}\}.$$
Note that $|M^{(q)}| = q$.
4) The set, $\{M^{(1)}, \ldots, M^{(|U|)}\}$, becomes our new search space in Eq. (6) so that the constraint, $M \subset \mathcal{S}$, becomes $M \in \{M^{(1)}, \ldots, M^{(|U|)}\}$. The minimizing candidate, M^*, becomes our greedy approximation to the Fréchet mean, μ.

While it is possible to further reduce the search space for M^* by starting with $M^{(|I|)}$, the candidate mean containing elements of the intersection, we do not find that it significantly improves computational performance in practice.

To apply the approximate Fréchet mean as the representative item $\bar{x}_j$, $j = 0, \ldots, m-1$, in the prediction function defined by Eq. (3), we note that the observed LZSets in Eq. (6) may be defined to be LZSets belonging to class j. Prior publications have considered alternative notions of a representative set for

[2]The Fréchet mean set is currently not constrained to be a proper LZSet.

a collection of observed sets based on the Jaccard distance [34]–[36]. The focus of our work, however, is not to obtain an exact representative set, as defined by the set which minimizes Eq. (5) or some other cost function. Instead, our objective is to obtain a representative set for making predictions according to Eq. (3). We will show that our greedy algorithm is both computationally efficient and produces a representative set that yields good prediction accuracies on a variety of datasets, which we describe next.

III. Data

We first describe a toy dataset used to validate the greedy approximation of the Fréchet mean described in Section II. We then describe datasets used to test the performance of the Fréchet mean for classification on authorship attribution of source code, file fragment type detection, and malware classification. The variety of datasets is meant to demonstrate the generality of our featureless approach on a range of computer security applications.

A. Toy dataset

The toy dataset consists of sequences constructed from a discrete random variable $c \in \{1, \ldots, r\}$ with specified entropy, h in bits, where h is defined by

$$h(c) = -\sum_{i=1}^{r} \Pr(c = i) \log_2 \Pr(c = i). \tag{8}$$

Thus, for a specified h, we numerically solve for the corresponding probability law of the r-valued random variable c, where the number of states is given by $r = \lceil 2^h \rceil$. We note that the solution is non-unique.

B. Java source code

The single-author open-source dataset described by Yang et al. in Ref. [37] is available online at [38]. The dataset contains 3,022 Java files ranging from 16 lines to 11,418 lines, with an average line length of about 99. Each Java file is attributed to one of forty authors; the minimum and maximum number of files contributed by the authors were 11 and 712, respectively.

C. File fragments

The Govdocs1 dataset of 1 million documents was collected with the goal of providing a more standardized dataset for digital forensics research [39]. Our file fragment dataset is derived from about 20% of the Govdocs1 files. Although there were many more file extensions and file sizes, we only used files with extensions of .doc, .gif, .html, .jpg, and .pdf and with size of at least 1 kilobyte. From this subset of files, we constructed one file fragment per file. Each file fragment consists of 512 contiguous bytes, chosen at random file offsets, where the offsets are at least 512 bytes from the end of the files.

D. Microsoft malware

The Microsoft Malware Classification Challenge [40] was developed as a Kaggle competition with the objective of classifying malware samples into one of nine families. Since 2015, it has been cited in more than 70 papers. The dataset contains approximately 10,000 labeled samples and approximately 10,000 unlabeled samples. For every sample the binary contents were represented as both a hexadecimal string and as the output from the IDA disassembler. In our study, we focus on the hexadecimal representation of the binary contents.

E. Drebin malware

The Drebin malware dataset consists of Android Package (Android APK) malware saved from August 2010 to October 2012. This dataset is publicly available and has become very popular in malware studies, having been cited over 1,500 times in publications. The dataset consists of 5,560 labeled samples belonging to 180 classes of malware. In our study, much like the original work studying this dataset [41], [42], we limit our study to the top-20 classes of malware, which amounts to 4,664 labeled samples (about 84% of the data). The reason for this downsampling is that many of the other classes had too few samples to learn enough for class-type inference. While the Android APK files are extended from the JAR file format and can hence be untarred, we limit our study to the original (APK) form of the data.

IV. Results

We first present results on the greedy approximation to the Fréchet mean. We then apply the approximated Fréchet mean for inference on the datasets described in Section III. Our method is compared with other prototype methods using the LZJD metric and, where possible, previously reported methods on the same datasets. We find that the Fréchet mean demonstrates comparable prediction accuracy and superior prediction efficiency across multiple applications considered, with none of the feature engineering overhead required by more traditional methods.

All experiments were executed on a 24-core Intel(R) Xeon(R) CPU E5-2695 clocked at 2.4 GHz with 768 GB of memory. The software used to perform these experiments were written in python3.

A. Greedy approximation to the Fréchet mean

To better understand the approximation of the greedy algorithm, we look at the shape of the cost defined in Eq. (5) as a function of $M^{(q)}$, for $q = 1, \ldots, |U|$. The toy dataset described in Section III-A is used to generate input sequences $\mathbf{x} = x_1, \ldots, x_{10}$ with specified source entropy h. Each input sequence in $\mathbf{x}$ has identical length of 10 and identical h.

Figure 3 shows the cost of the rank-ordered candidate means; different curves correspond to different entropies for the input sequences. In all cases, the cost function appears quasiconvex and the global minimum corresponding to the approximate Fréchet mean, M^*, is indicated by the marker. With increasing entropy, several observations can be made. In

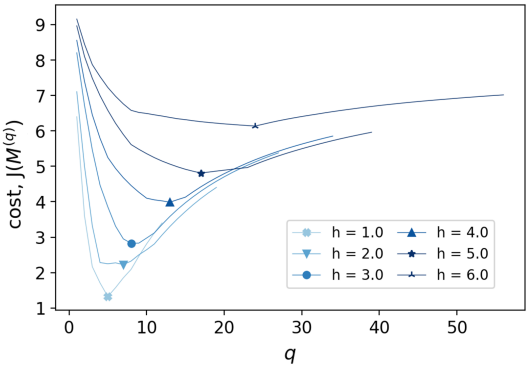

Fig. 3. Cost of the candidate mean $M^{(q)}$ as a function of its index q obtained using the greedy algorithm. Toy dataset consists of input sequences $\mathbf{x}$ generated using a random source with specified entropy h.

general, the overall cost function shifts towards higher costs, indicating a trend towards increased distances between the candidate means and the input sets; the maximum value for q increases, indicating that the size of the search space also increases; and finally, the cost and size of M^* also increases.

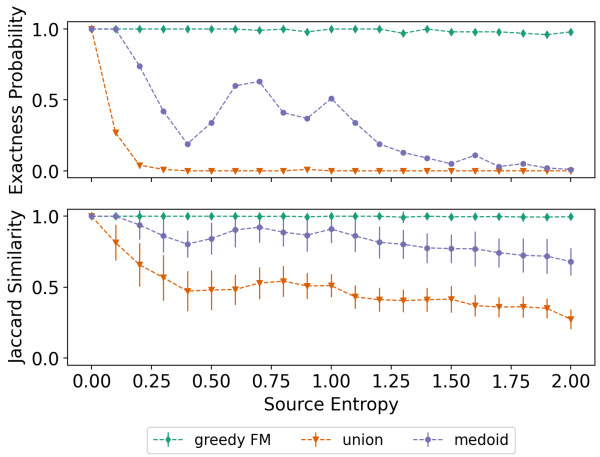

Fig. 4. Comparing the representative sets with the true Fréchet mean calculated from brute-force; see Section IV-A for a definition of the *exactness probability* and the *Jaccard similarity*. Toy dataset consists of input sequences generated using a random source with specified entropy h. Mean and standard deviation are calculated over 100 runs.

Next, using the same toy dataset as described for Fig. 3, we investigate how close M^* is to the true Fréchet mean, μ. We consider a brute-force search for μ in Eq. (6) over the sets that are both superset of the intersection and subset of the union. Due to the computational cost of the brute-force calculation for μ, only entropies $h \leq 2$ are considered. To quantify the closeness of M^* to μ, we calculate the *exactness probability*, or the empirical probability that M^* matches μ, and the average *Jaccard similarity*, defined by $|M^* \cap \mu|/|M^* \cup \mu|$, across 100 different runs.

Figure 4 shows the exactness probability (top) and the Jaccard similarity (bottom) as a function of the source entropy. Here, in addition to M^*, we introduce the union and the medoid as alternative representative sets to compare with μ. The medoid is similar in concept to the Fréchet mean, but is restricted to be a member of the observed data items. According to both measures, the greedy Fréchet mean M^* is a very good approximation of μ for all entropies considered. However, the same is not true for the union or the medoid.

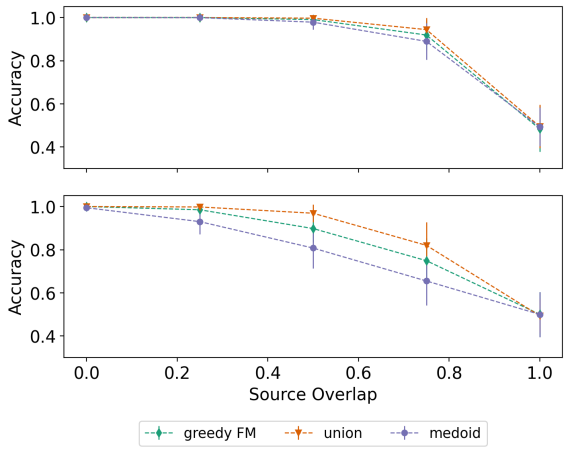

Fig. 5. Accuracy of binary classification using the representative sets. Toy dataset consists of input sequences generated using a random source with specified entropy $h = 2$ (top) and $h = 4$ (bottom). Mean and standard deviation are calculated over 100 runs, where each run performs predictions on 10 items selected per class.

Having quantified how close the different representative sets are to the true Fréchet mean μ, it remains to apply them in a classification problem using the LZJD metric. We consider a simple binary classification problem using the same toy dataset with specified source entropy h. To differentiate between the two classes, we control the overlap in the alphabet between the two sources. Overlap is defined as the ratio of the cardinalities of the intersection to the alphabet set ($|A_1 \cap A_2|/|A_1|$ and $|A_1| = |A_2|$). For an overlap of 0.0, the alphabets for the two classes are disjoint and the classification task is trivial; for an overlap of 1.0, the alphabets for the two classes are identical and the classification task is expected to perform random.

Figure 5 shows the accuracy as a function of source overlap for $h = 2$ (top) and $h = 4$ (bottom). In general, accuracy decreases with increasing overlap and with increasing entropy. Although we have provided a simple example involving a toy dataset, it can be seen that how close a representative set is to the true Fréchet mean μ does not always determine how good it performs at a given classification task. In particular, note that the union outperforms both M^* and the medoid at classification in Fig. 5, but is the farthest from μ, as quantified by both the exactness probability and the Jaccard similarity in Fig. 4. However, on the real datasets considered in this study, we will see that M^* consistently outperforms both the union and the medoid.

B. Authorship attribution

Authorship attribution refers to the task of attributing a piece of text to a predefined set of authors. Authorship attribution

Method	Accuracy (%)	Balanced Accuracy (%)	total t_{train}, (s)	per item t_{pred}, (s)
Greedy Fréchet mean	90.4 (2.1)	87.7 (2.3)	5.5×10^{1} (6.9×10^{-1})	5.1×10^{-3} (1.8×10^{-4})
Union	11.8 (2.2)	72.6 (6.5)	6.3×10^{-1} (1.2×10^{-2})	2.6×10^{-2} (2.6×10^{-4})
Medoid	83.0 (1.4)	75.8 (2.2)	3.3×10^{1} (8.4×10^{-1})	3.5×10^{-3} (1.2×10^{-4})
Average distance	87.7 (1.9)	84.4 (2.6)	1.3×10^{-3} (1.6×10^{-4})	1.3×10^{-1} (2.4×10^{-3})
1-NN	96.4 (0.9)	94.2 (1.9)	1.2×10^{-3} (1.4×10^{-4})	1.3×10^{-1} (2.6×10^{-3})

Table I. Authorship attribution of source code. Italicized methods are based on representative sets. Entries show mean and standard deviation (in parentheses) over a 10-fold cross-validation experiment, where we train on large fold.

of source code has several useful applications, including detecting plagiarism, resolving authorship disputes, tracking malicious code, and software forensics. The most successful demonstrations of author attribution of source code have largely relied on derived feature sets; see [43] for an overview. Herein we demonstrate successful authorship attribution using the featureless LZJD metric.

Table I shows prediction and timing performance results for authorship attribution. Here, and in all subsequent experiments, we first extracted the LZSets for each data item. Since all prototype methods based on the LZJD metric depend on LZSets, we do not include LZSet creation time in either the training or prediction times. Italicized methods are based on distances to *representative* sets. Similar to [44], we show the accuracy, defined by

$$\text{accuracy} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(\hat{y}_i = y_i),$$

on a 10-fold cross-validation experiment, where we train on the large fold. However, due to significant class imbalance, we also present the balanced accuracy, defined as

$$\text{balanced accuracy} = \frac{1}{m}\sum_{j=1}^{m}\left[\frac{1}{|\mathcal{X}_j|}\sum_{x_i \in \mathcal{X}_j} \mathbf{1}(\hat{y}_i = j)\right].$$

In this expression, recall that $\mathcal{X}_j \subset \mathcal{X}$ is the subset of $\mathcal{X}$ whose items x_i have corresponding label $y_i = j$. Note that the balanced accuracy is equivalent to the average recall on each class.

The 1-NN classifier achieved both the highest accuracy (96.4%) and balanced accuracy (94.2%). Larger number of neighbors (k) were also tried, but resulted in poorer classification performance, likely due to the significant class imbalance. The accuracy of the 1-NN classifier was followed by our greedy approximation to the Fréchet mean, and, in decreasing (balanced) accuracy, the average distance, medoid, and union classifiers. Although our goal is not to outperform previous methods in classification accuracy, per se, it is still impressive that the top performer using the LZJD metric is comparable to the path-based methods described by [44], which are, to the best of our knowledge, the highest performers on this dataset.

Timing results indicate that the methods based on representative sets (italicized) are 1–2 orders of magnitude faster at inference, with the union requiring slightly more time due to the growth in the size of the union set. Note, however, that the improved inference time comes at a cost. The time required to compute the Fréchet mean or medoid from the observed data items increases their training time considerably, as compared with the methods based on observed sets. Specifically, training for the methods based on observed sets essentially consists of creating the classes of (precalculated) LZSets so that the difference in time can be attributed to calculating the representative set. Although the union does not have the same upfront increase in training time compared with the other representative sets, we note that it has the lowest (balanced) accuracy across all methods considered.

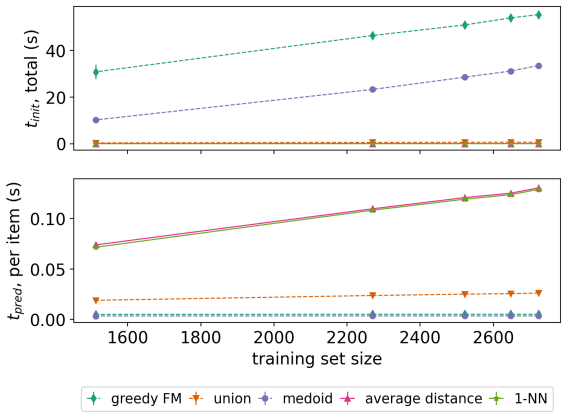

Fig. 6. Authorship attribution of source code. Timing results on the training and prediction steps, as a function of the number of items in the training set.

Figure 6 shows the total training times and per item predict times as a function of the number of items in the training set. The number of items in the training set is dictated by the number of folds in the cross fold validation experiment; 2-, 4-, 6-, 8-, and 10-fold cross-validation results are presented. Since we train on the large fold, the size of the training set increases with the number of folds. Note that timing results for the 10-fold cross-validation are also presented in Table I. Although calculation of the greedy Fréchet mean and the medoid requires upfront compute time that increases with the size of the training set (Fig. 6, top), it can be seen that their per item predict times are independent of the size of the training set (Fig. 6, bottom).

C. File fragment classification

The fragmentation of files poses a significant challenge for data recovery. File carving refers to the task of reconstructing complete files from the content of file fragments. A key task in file carving is file fragment classification, where the objective

Method	Accuracy (%)	total t_{train}, (s)	per item t_{pred}, (s)
Greedy Fréchet mean	59.1 (0.9)	6.1×10^{1} (5.5×10^{-1})	2.2×10^{-1} (8.3×10^{-6})
Union	35.3 (1.2)	1.8×10^{-1} (6.2×10^{-3})	7.9 (2.1×10^{-4})
Medoid	50.1 (1.6)	1.6×10^{1} (4.4×10^{-1})	2.0×10^{-1} (2.9×10^{-6})
Average distance	52.0 (2.3)	1.1×10^{-3} (7.9×10^{-5})	3.2×10^{1} (8.6×10^{-4})
1-NN	58.7 (1.2)	1.3×10^{-3} (1.5×10^{-4})	3.2×10^{1} (5.1×10^{-4})
9-NN	60.8 (1.3)	1.1×10^{-3} (1.8×10^{-4})	3.1×10^{1} (4.5×10^{-4})

Table II. File fragment classification. Italicized methods are based on representative sets. Entries show mean and standard deviation (in parentheses) obtained over 10 runs.

		Predicted				
		DOC	GIF	HTML	JPG	PDF
Actual	DOC	**56.6**	18.5	21.2	3.7	0.1
	GIF	0.4	**97.3**	0.0	2.2	0.1
	HTML	0.4	0.0	**99.6**	0.0	0.0
	JPG	1.3	58.3	1.2	**39.0**	0.2
	PDF	0.7	76.9	14.3	5.3	**2.8**

		Predicted				
		DOC	GIF	HTML	JPG	PDF
Actual	DOC	**77.6**	2.4	2.6	11.9	5.5
	GIF	0.8	**16.7**	0.0	56.1	26.3
	HTML	8.7	0.0	**90.3**	0.0	0.9
	JPG	1.1	5.9	0.2	**71.6**	21.2
	PDF	0.4	6.4	1.0	44.3	**47.8**

Table III. File fragment classification. Normalized confusion matrices of **greedy Fréchet mean** (left) and **9-NN** (right) classifiers. Entries correspond to mean values over 10 runs.

is to determine the type of file a fragment belongs to based on its content. Prior work in file fragment classification has primarily relied on hand-crafted features. In [45], Wang et al. propose an automated method for feature extraction based on sparse coding, or sparse dictionary learning; the extracted features contained in the dictionary are based on how well those features can be used to reconstruct the original data. The sparse dictionary is conceptually similar to the lookup table of a compression dictionary: when a new token is observed, a search through the dictionary for the best match that is sparse, i.e., compressible, is made.

Table II presents prediction and timing performance results for the different prototype methods. Once again, italicized methods are based on representative sets. Similar to [45], our experiments use a balanced 450 training to 300 testing ratio per class, and results are averaged over 10 independent runs. Because we use a balanced dataset of file fragments, we only show the total accuracy. The k-NN and greedy Fréchet mean classifiers yield the highest accuracies, near 60% and within one standard deviation of each other. With comparable prediction accuracies to the highest performers, the greedy Fréchet mean is also two orders of magnitude faster during inference. Precisely how much faster will depend on the number and size of the file fragments in the training set. As before, faster inference comes at the cost of slower training necessary for computing the Fréchet mean.

To better understand the source of the observed accuracies, Table III shows the normalized confusion matrix for the greedy Fréchet mean and the 9-NN classifiers. Although it is difficult to make a direct comparison with [45], we observe a similar behavior in the file types that are most often and least often misclassified. That is, file types that contain identifying patterns, such as open and closed brackets in `HTML`, are the most easily classified. At the other end, file types that have some compression in their encoding (`JPG`, `GIF`), or file types that contain other files (`PDF`) are more difficult to classify. In particular, the greedy Fréchet mean classifier tends to classify `GIF`, `JPG`, and `PDF` as all `GIF`, whereas the 9-NN tends to classify them as either `JPG` or `PDF`.

D. Malware

For both malware classification problems, the objective is to identify which class of malware a particular file belongs to. In the Microsoft malware dataset, these are executables designed for Microsoft systems with the PE File Headers removed and the classes are one of nine identified families. In the Drebin malware dataset, these are Android APKs and the classes are the twenty most common families of malware in the dataset. Both datasets have been studied with a variety of different classification techniques; here we compare the Fréchet mean against the 1-NN and average distance classifiers.

Microsoft malware: The sequences of malware files sometimes exceed two million bytes in length, so for the sake of computational time we selected a random 2% and 10% subset of the data. Additionally, since all methods required calculating LZSets from the input data, we performed that operation once per experiment and did not include those timings in the tables. For reference, it took 1,100 (7,100) seconds to create LZSets for 2 (10)% of the data.

We capture accuracy, balanced accuracy, and timing performance in Tables IV and V for the 2% and 10% subsets, respectively. In order to compare our prediction accuracies to [25] we ran 10-fold cross-validation on each of the two subsets. The greedy Fréchet mean algorithm yields the best accuracy on both subsets of the data, but does not perform as well as the 1-NN algorithm in class balanced accuracy. The greedy Fréchet mean cannot provide as complex of a decision boundary as a 1-NN algorithm. Furthermore, since this dataset has a large class imbalance, the ability to generate an appropriate representative set for all classes may be hindered. Even

Method	Accuracy (%)	Balanced Accuracy (%)	total t_{train}, (s)	total t_{pred}, (s)
Greedy Fréchet mean	73.0 (13.3)	67.1 (13.7)	2.0×10^3 (8.4×10^1)	5.0 (4.0×10^{-1})
Average distance	59.5 (11.3)	60.8 (12.1)	3.8×10^{-4} (3.6×10^{-5})	2.9×10^1 (3.0)
1-NN	61.5 (9.2)	72.1 (6.2)	2.7×10^{-4} (1.8×10^{-5})	3.0×10^1 (3.2)

Table IV. Microsoft malware family classification on **2% of data**. 10-fold cross-validation.

Method	Accuracy (%)	Balanced Accuracy (%)	total t_{train}, (s)	total t_{pred}, (s)
Greedy Fréchet mean	79.5 (7.2)	76.1 (9.6)	1.5×10^4 (1.2×10^3)	3.0×10^1 (2.5)
Average distance	27.9 (4.3)	36.2 (5.6)	8.4×10^{-4} (3.9×10^{-5})	7.3×10^2 (9.1×10^1)
1-NN	77.5 (6.2)	85.9 (3.8)	7.7×10^{-4} (6.5×10^{-5})	7.2×10^2 (8.8×10^1)

Table V. Microsoft malware family classification **on 10% of data**. 10-fold cross-validation.

so, the class balanced accuracy of the greedy Fréchet mean is within one standard deviation of the 1-NN algorithm. For reference, using the 1-NN classifier on 10% of the data, [25] observed 58.1% class balanced accuracy for NCD and 98.2% for LZJD; we could not determine which data subset yielded those results.

We can see from the tables that the greedy Fréchet mean is significantly faster during inference than the prototype methods that require comparisons to all observed LZSets. This comes with the penalty of computing the Fréchet mean during training. For the average distance and 1-NN classifiers, the reported training time corresponds to creating the classes of (precalculated) LZSets so that the difference in training time is essentially the time required to compute the Fréchet mean using our greedy algorithm.

Drebin malware: Similar to the Microsoft malware, results for the Drebin malware dataset were generated by offline computation of the LZSets followed by 10-fold cross-validation over 2% and 10% of the dataset. The results for the Fréchet mean along with the 1-NN and average distance classifiers are presented in Table VI and Table VII for the 2% and 10% subsets, respectively.

For the Drebin dataset, the 1-NN had better accuracy when compared to the approximate Fréchet mean. This is possibly due to the smaller dataset, which makes it more difficult to compute a good representative set because some classes have very few observed items. In particular, 11 of the 20 malware classes had a total of 734 observed items as compared to 925 of the modal class. This means that for the experiment on 2% of the data, we expect worse performance for the representative set as many classes have too few observed items to use in approximating the Fréchet mean (Table VI). On the other hand, as we increased to 10% of the data, we gained better approximations for the less frequent malware classes and hence better classification accuracy (Table VII). We note that results for the 1-NN were also presented in [25] as the "Drebin APK" LZJD. Our results for the 1-NN differ, but that is likely due to different random choices of sequences.

As shown in the tables, besides the computation of the LZSets, the additional training times for the 1-NN and average distance methods were almost zero. This is because the training is simply saving the observed LZSets in memory. The creation of the LZSets, which was done offline, took about 94 seconds for 2% of the data and 567 seconds for 10% of the data. Contrary to the training period, during inference, the Fréchet mean method mostly outperformed in running time. This is because the comparisons for inference only used one item, the representative Fréchet mean, per class. On the other hand, the inference for the 1-NN and average distance methods required comparisons per observed item per class. When the experiment was on 2% of the data, we suspect that the size of the Fréchet mean was large enough to still be slightly slower than when comparing with the relatively few samples per class as shown in Table VI.

V. Conclusions

Compression analytics are a class of featureless machine learning methods that have demonstrated promise in computer security applications. We have presented a method that, in addition to being featureless, performs fast and accurate inference that is independent of the size of each labeled class. This is achieved by calculating a representative sample, the greedy Fréchet mean, for each labeled class and applying it at prediction time. We have demonstrated, across a variety of applications, that our method is generally faster and provides similar classification performance when compared with previous methods that require the full training set at inference, e.g., k-NN. Fast featureless classification algorithms such as that presented here allow researchers and analysts to address a wider range of problems on a wider variety of devices than previously possible.

Although the objective of this work was faster prediction, faster training is still of interest. As mentioned at the end of Section II-A, it is possible to use hashed LZSets to improve the speed of every algorithm considered here, particularly the calculation of the greedy Fréchet mean. Furthermore, our implementations of all the algorithms are fairly naive and many are embarrassingly parallel. These, and other implementation improvements, will be the subject of future work.

Acknowledgment

The authors thank Rich Field for providing the code used to generate the toy dataset; and Erin Acquesta and Rich

Method	Accuracy (%)	Balanced Accuracy (%)	total t_{train}, (s)	total t_{pred}, (s)
Greedy Fréchet mean	40.9 (14.2)	38.5 (13.7)	1.5×10^3 (4.6×10^1)	1.2×10^1 (1.3)
Average distance	32.1 (11.1)	32.4 (12.7)	4.6×10^{-4} (4.5×10^{-5})	8.7 (1.0)
1-NN	66.4 (16.8)	62.5 (18.7)	3.4×10^{-4} (2.2×10^{-5})	8.5 (8.9×10^{-1})

Table VI. Drebin malware family classification on **2% of data**. 10-fold cross-validation.

Method	Accuracy (%)	Balanced Accuracy (%)	total t_{train}, (s)	total t_{pred}, (s)
Greedy Fréchet mean	57.7 (7.5)	52.2 (10.1)	7.3×10^3 (9.9×10^1)	9.2×10^1 (5.8)
Average distance	40.1 (9.5)	43.5 (10.9)	6.9×10^{-4} (1.7×10^{-5})	2.6×10^2 (1.3×10^1)
1-NN	78.7 (8.3)	73.8 (10.4)	5.7×10^{-4} (2.9×10^{-5})	2.7×10^2 (1.3×10^1)

Table VII. Drebin malware family classification on **10% of data**. 10-fold cross-validation.

Field of Sandia National Labs for reviewing this paper and providing helpful feedback for improving our presentation of this work. The authors also acknowledge Travis Bauer of Sandia National Labs, who helped inspire this work with useful initial discussions on this topic. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND2020-xxxx.

REFERENCES

[1] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, 2001, vol. 1, no. 10.

[2] L. Yang and R. Jin, "Distance metric learning: A comprehensive survey," *Michigan State Universiy*, vol. 2, no. 2, p. 4, 2006.

[3] I. Kirillov, D. Beck, P. Chase, and R. Martin, "Malware attribute enumeration and characterization," *The MITRE Corporation [online, accessed Apr. 8, 2019]*, 2011.

[4] A. Singh, A. Walenstein, and A. Lakhotia, "Tracking concept drift in malware families," in *Proceedings of the 5th ACM workshop on Security and artificial intelligence*, 2012, pp. 81–92.

[5] W. P. Kegelmeyer, K. Chiang, and J. Ingram, "Streaming malware classification in the presence of concept drift and class imbalance," in *2013 12th International Conference on Machine Learning and Applications*, vol. 2. IEEE, 2013, pp. 48–53.

[6] R. Jordaney, K. Sharad, S. K. Dash, Z. Wang, D. Papini, I. Nouretdinov, and L. Cavallaro, "Transcend: Detecting concept drift in malware classification models," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 625–642.

[7] O. De Vel, A. Anderson, M. Corney, and G. Mohay, "Mining e-mail content for author identification forensics," *ACM Sigmod Record*, vol. 30, no. 4, pp. 55–64, 2001.

[8] D. Ariu, G. Giacinto, and F. Roli, "Machine learning in computer forensics (and the lessons learned from machine learning in computer security)," in *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, 2011, pp. 99–104.

[9] V. Roussev and S. L. Garfinkel, "File fragment classification-the case for specialized approaches," in *2009 Fourth international IEEE workshop on systematic approaches to digital forensic engineering*. IEEE, 2009, pp. 3–14.

[10] P. M. Vitányi, F. J. Balbach, R. L. Cilibrasi, and M. Li, "Normalized information distance," in *Information theory and statistical learning*. Springer, 2009, pp. 45–82.

[11] M. Li, P. Vitányi *et al.*, *An introduction to Kolmogorov complexity and its applications*. Springer, 2008, vol. 3.

[12] M. Li, X. Chen, X. Li, B. Ma, and P. M. Vitányi, "The similarity metric," *IEEE transactions on Information Theory*, vol. 50, no. 12, pp. 3250–3264, 2004.

[13] R. Cilibrasi and P. M. Vitányi, "Clustering by compression," *IEEE Transactions on Information theory*, vol. 51, no. 4, pp. 1523–1545, 2005.

[14] E. Stamatatos, "A survey of modern authorship attribution methods," *Journal of the American Society for information Science and Technology*, vol. 60, no. 3, pp. 538–556, 2009.

[15] A. Bardera, M. Feixas, I. Boada, and M. Sbert, "Compression-based image registration," in *2006 IEEE International Symposium on Information Theory*. IEEE, 2006, pp. 436–440.

[16] S. Wehner, "Analyzing worms and network traffic using compression," *Journal of Computer Security*, vol. 15, no. 3, pp. 303–320, 2007.

[17] C. Ting, R. Field, A. Fisher, and T. Bauer, "Compression analytics for classification and anomaly detection within network communication," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 5, pp. 1366–1376, 2018.

[18] J. S Resende, R. Martins, and L. Antunes, "A survey on using Kolmogorov complexity in cybersecurity," *Entropy*, vol. 21, no. 12, p. 1196, 2019.

[19] A. Paturi, M. Cherukuri, J. Donahue, and S. Mukkamala, "Mobile malware visual analytics and similarities of attack toolkits (malware gene analysis)," in *2013 International Conference on Collaboration Technologies and Systems (CTS)*. IEEE, 2013, pp. 149–154.

[20] R. S. Borbely, "On normalized compression distance and large malware," *Journal of Computer Virology and Hacking Techniques*, vol. 12, no. 4, pp. 235–242, 2016.

[21] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario, "Automated classification and analysis of internet malware," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2007, pp. 178–197.

[22] A. Macedonas, D. Besiris, G. Economou, and S. Fotopoulos, "Dictionary based color image retrieval," *Journal of Visual Communication and Image Representation*, vol. 19, no. 7, pp. 464–470, 2008.

[23] D. Cerra and M. Datcu, "A fast compression-based similarity measure with applications to content-based image retrieval," *Journal of Visual Communication and Image Representation*, vol. 23, no. 2, pp. 293–302, 2012.

[24] H. Koga, Y. Nakajima, and T. Toda, "Effective construction of compression-based feature space," in *2016 International Symposium on Information Theory and Its Applications (ISITA)*. IEEE, 2016, pp. 116–120.

[25] E. Raff and C. Nicholas, "An alternative to NCD for large sequences, Lempel-Ziv Jaccard distance," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 1007–1015.

[26] C. Ting, R. Field, T.-T. Quach, and T. Bauer, "Generalized boundary detection using compression-based analytics," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3522–3526.

[27] S. Kosub, "A note on the triangle inequality for the Jaccard distance," *Pattern Recognition Letters*, vol. 120, pp. 36–38, 2019.

[28] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[29] B. Gunsel, A. K. Jain, A. M. Tekalp, and B. Sankur, *Multimedia Content Representation, Classification and Security: International Workshop, MRCS 2006, Istanbul, Turkey, September 11-13, 2006, Proceedings*. Springer, 2006, vol. 4105.

[30] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on information theory*, vol. 23, no. 3, pp. 337–343, 1977.

[31] ——, "Compression of individual sequences via variable-rate coding," *IEEE transactions on Information Theory*, vol. 24, no. 5, pp. 530–536, 1978.

[32] F. Nielsen and R. Bhatia, *Matrix Information Geometry*. Springer, 2013.

[33] E. Raff and C. Nicholas, "Lempel-Ziv Jaccard distance, an effective alternative to ssdeep and sdhash," *Digital Investigation*, vol. 24, pp. 34–49, 2018.

[34] H. Späth, "The minisum location problem for the Jaccard metric," *Operations-Research-Spektrum*, vol. 3, no. 2, pp. 91–94, 1981.

[35] F. Chierichetti, R. Kumar, S. Pandey, and S. Vassilvitskii, "Finding the Jaccard median," in *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*. SIAM, 2010, pp. 293–311.

[36] M. Bury and C. Schwiegelshohn, "On finding the Jaccard center," in *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[37] X. Yang, G. Xu, Q. Li, Y. Guo, and M. Zhang, "Authorship attribution of source code by using back propagation neural network based on particle swarm optimization," *PloS one*, vol. 12, no. 11, p. e0187204, 2017.

[38] "40 authors java source code dataset," https://github.com/xinyu1118/authorship_attribution, accessed: 2020-07-22.

[39] "Digital corpora govdocs1 dataset," https://digitalcorpora.org/corpora/files, accessed: 2018-08-17.

[40] R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, and M. Ahmadi, "Microsoft malware classification challenge," *CoRR*, vol. abs/1802.10135, 2018.

[41] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket." in *NDSS*, vol. 14, 2014, pp. 23–26.

[42] S. Michael, E. Florian, S. Thomas, C. F. Felix, and J. Hoffmann, "Mobilesandbox: Looking deeper into android applications," in *Proceedings of the 28th International ACM Symposium on Applied Computing (SAC)*, 2013.

[43] V. Kalgutkar, R. Kaur, H. Gonzalez, N. Stakhanova, and A. Matyukhina, "Code authorship attribution: Methods and challenges," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1–36, 2019.

[44] E. Bogomolov, V. Kovalenko, A. Bacchelli, and T. Bryksin, "Authorship attribution of source code: A language-agnostic approach and applicability in software engineering," *arXiv preprint arXiv:2001.11593*, 2020.

[45] F. Wang, T.-T. Quach, J. Wheeler, J. B. Aimone, and C. D. James, "Sparse coding for n-gram feature extraction and training for file fragment classification," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 10, pp. 2553–2562, 2018.