Statistical detection of format dialects using the weighted Dowker complex

Michael Robinson
Department of Mathematics and Statistics
American University
Washington, DC
Email: michaelr@american.edu

Cory Anderson
BAE Systems FAST Labs
Arlington, VA
Email: cory.s.anderson@baesystems.com

Steve Huntsman
Arlington, VA
Email: sch213@nyu.edu

Letitia W. Li
BAE Systems FAST Labs
Arlington, VA
Email: letitia.li@baesystems.com

***Abstract*—This paper provides an experimentally validated, probabilistic model of file behavior when consumed by a set of pre-existing parsers. File behavior is measured by way of a standardized set of Boolean "messages" produced as the files are read. By thresholding the posterior probability that a file exhibiting a particular set of messages is from a particular dialect, our model yields a practical classification algorithm for two dialects. We demonstrate that this thresholding algorithm for two dialects can be bootstrapped from a training set consisting primarily of one dialect. Both the theoretical and the empirical distributions of file behaviors for one dialect yield good classification performance, and outperform classification based on simply counting messages.**

Our theoretical framework relies on statistical independence of messages within each dialect. Violations of this assumption are detectable and allow a format analyst to identify "boundaries" between dialects. By restricting their attention to the files that lie within these boundaries, format analysts can more efficiently craft new criteria for dialect detection.

I. Introduction

This paper provides an experimentally validated, probabilistic model of file behavior on a set of Boolean features ("messages"). By thresholding the posterior probability that a file exhibiting a particular set of messages is from a particular dialect, our model yields a practical classification algorithm for two dialects. We demonstrate that this thresholding algorithm for two dialects can be bootstrapped from a training set consisting primarily of one dialect. Both the theoretical and the empirical distributions of file behaviors for one dialect yield good classification performance. Furthermore, although message count may correlate with one dialect over another, we show that our approach yields a substantially better classifier because it correctly dispositions the files that do not follow this trend.

Although our theoretical framework relies on statistical independence of messages within each dialect, violations of this assumption are detectable and allow a format analyst to identify "boundaries" between dialects. The key payoff of this approach is that we can tell which message patterns are easy to disposition as one dialect or another, and we can also identify which message patterns are ambiguous. This information allows an analyst to pinpoint specific features that account for the files on which two (or more) parsers disagree.

A. Application context

File format specifications are dynamic entities, and are often ambiguous. A given clause in a specification may have several distinct but self-consistent interpretations, and these interpretations may impact the interpretations of other, related clauses. As a result, files from different dialects of a format tend to exhibit divergent behavior at multiple, independent points within a parser's code base. Our methodology exploits this independence structure to discriminate between dialects.

When standards organizations attempt to resolve an ambiguity in the specification, stakeholders bring example files that exhibit specific behaviors. One may suspect that these files are not unbiased samples from a statistical perspective! Presently, there appears to be no unbiased way to query the corpus of files "in the wild" to find examples of files whose behavior is ambiguous. Our methodology provides an even-handed, systematic way to use an existing set of parsers and a large corpus of files to identify specific parser message patterns that are either easy (or difficult) to disposition as one dialect or another. Filtering for these message patterns can enable an expert user to identify a sample of files that are representative of the dominant dialects with different interpretations of the specification.

B. Background

Beyond what our team has previously published in the past year or so [1], [2], there appears to be very little work in analyzing file behavior using statistical tools. In contrast, nearly all existing file format analysis uses the structure of file

contents rather than the responses of parsers to those contents (for instance, see [3]–[7]).

Most relevant to our work, [8] notes that 39 valid dialects for CSV exist, which makes parsing any given CSV file challenging. Dialects all contain their own set of quotation marks, escape characters, delimiters, headers, comment text, etc. Their approach identifies the dialect of a file by using consistency measures that are expected to score higher if the dialect is identified. When the dialect and its associated delimiters are used correctly in a parse, the number of cells in a row should all be the same, and the types of cells in a single column should all be identical.

In a related vein, the PADS project aids users by generating a formal language description of an *ad hoc* format, including a inference algorithm that uses statistical histograms of tokens to identify if tokens should be combined into arrays or structures [9]. The PADS methodology is similar to ours, in that a collection of simple filters are deployed against the data, from which relatively sparse responses are collected. One proceeds to identify unusual patterns of responses across the ensemble of filters, much as we do. However, while PADS attempts to identify the underlying datastructures from this statistical inference, we are more interested in understanding whether a particular format dialect is being used.

Statistical analysis of files has also been used for malicious file detection or analysis of collected drives as digital forensic evidence. The DIRIM tool detects suspicious files or drives based on file metadata [10]. It uses PCA and k-means to cluster files and determine features more likely indicative of files of interest, such as those attempting to conceal their file type with a misleading file extension. Statistical features based upon file actions has also been used to identify certain malicious behaviors [11]. Although they start with data that are formatted similarly to ours, their ultimate goal is simply classification rather than dialect identification.

In other applications, clustering has been used to identify natural language dialects. Lundberg showed that one can relate clusters in recordings from Swedish speakers to spoken dialects. Recordings were converted into acoustic features, and clustered using PCA, k-means, and hierarchical clustering [12]. Grieve *et al.* also used hierarchical clustering for classification of dialects across different regions of the US, based on use of lexical alternation variables (for instance, "actually" versus "in fact") [13].

Finally, we note in passing the structural similarity between the weighted Dowker complex approach used here and *factor analysis* [14]. Factor analysis effectively works from the opposite perspective to ours. One starts with an assumed dependence between variables rather than locating violations of independence when they occur in the data. The difference is important; violations of independence occur near "dialect boundaries" within a dataset. However, unlike factor analysis, which exploits an assumed dependence structure, our Dowker complex construction is mathematically well-defined regardless of any dependence structure within the data.

C. Assumptions and Limitations

This paper assumes that there is a pre-existing collection of Boolean "messages" produced by several parsers, and that each file under consideration has been processed by each parser. We will assume that these messages cover all of the relevant aspects of the dialects that we wish to measure, and that the messages are diverse enough to discriminate between these aspects. Our methodology is format agnostic, in that it does not look at the file contents directly. File contents are only considered through the lens of the pre-existing parsers, so a user of our method will not need to be a format expert. In our previous work [1], "messages" were based on the content of `stderr`. In that case, each message was best understood as an *error* message. In this article, we do not assume that a given message is related to an adverse condition. For instance, a message could be the presence or absence of a certain byte sequence in the file, or it may simply report whether the exit code for a given parser corresponds to a valid parse.

We note that the use of Boolean messages does not limit the expressiveness of our approach very much. Any finite collection of messages that are (partially) ordered, for instance messages coded as integers, can be losslessly encoded as a pattern of Boolean messages. Moreover, this encoding does not limit the effectiveness of our approach at all. (A theoretical exposition of these facts is currently in preparation.) We therefore do not need to consider non-Boolean messages in our approach.

The theoretical justification for our method relies upon the independence of messages for files within a given dialect. In a representative sample of messages for the dataset we describe in the next section, we found that most pairs of messages are independent, though a small subset of messages are highly dependent on other messages (see Section II-B). Even absent this justification, the weighted Dowker complex construction we give is mathematically well-defined, and so our algorithmic approach can still be followed.

In our specific dataset, this dependence arises because several parsers can be run with different options. Running the same parser with different options sometimes results in nearly identical messages being produced.

The proper strategy from the perspective of a formal model would be to capture the dependence structure, but it is *extremely* computationally infeasible to test for dependence (even pairwise dependence) whenever there are more than a few dozen messages. When we incorrectly assume conditional independence across all messages—as we will blithely proceed to do—this artificially reduces the probability of certain patterns of messages below what is actually observed. We can compensate for this effect by raising the overall message probability above what is estimated on a per-message basis. This appears to result in a distribution of message patterns that agrees with the observations, though some of the multi-way dependence structure is lost.

Finally, we will assume that for each dialect that we wish to study, there are only two kinds of messages: those that

occur frequently for that dialect, and those that occur at about the same frequency as for other dialects. Although apparently limiting, our dataset agrees with this assumption (see Figure 7). The less frequent kind of message is effectively a "background" message. At the start of our analysis, we do not know which message plays which role for any given dialect.

II. Dataset description

The data we processed to test our methodology were produced as training data by the Test and Evaluation Team for the DARPA SafeDocs evaluation exercise 3. These data consist of PDF files, ostensibly compliant with the ISO 32000-2 standard. For this exercise, we used the "Universe A" *good files* and *bad files* datasets. Each dataset consisted of 100001 hand-curated PDF files, for a total of 200002 files.

Given that the Test and Evaluation Team consists of PDF format experts, the Test and Evaluation Team was able to manually ensure that these two datasets had known ground truth: the good files are either syntactically and semantically valid PDF files, or are files that could be unambiguously corrected. The bad files exhibit various kinds of malformations including syntax errors, semantic violations, or other kinds of problems. The good files were largely sourced from Common Crawl [15], while the bad files were drawn from various sources: some were found in the wild (from Common Crawl), some were malicious files created by the Test and Evaluation Team, and others were non-malicious non-compliant files created by the Test and Evaluation Team.

Each file was processed through 13 distinct base parsers, run with various options to make a total of 29 parsers. A total of $\#M = 3022$ Boolean messages were collected, as shown in Table I. One message per parser is an exit code corresponding to the presence of an error, which accounts for a total of 29 messages. The rest of the messages correspond to specific regular expressions (regexes) run against `stderr`, as explained in Section II-A. Several of these messages were found to play an important role in identifying dialects and are discussed in detail in latter sections of this paper. The reader interested in seeing example regexes should consult Table V.

The input to the methodology described in this paper is therefore an unordered list of file-message pairs, recording the set of messages that occurred for each file. These data can be rendered into a matrix form, in which the rows correspond to messages and the columns correspond to files. The entries are either 1 or 0, if the message occurred or did not occur, respectively. The matrices for both datasets are shown in Figure 1. Since the same set of messages was collected for both datasets, the rows have the same meaning in both matrices. Even though the same number of files was present in each dataset, the meaning of the columns differs since the sets of files differ.

It is immediately clear visually that the two matrices are quite different. In particular, the bad files produce far more messages than the good files on average. A rough classification of an unknown file from one of these two sets based on its message count can clearly be an effective strategy. However, as will be shown in Section IV-D, our method outperforms this naïve strategy by a wide margin. One can determine which specific message patterns correspond to different behaviors, yielding a finer classification.

TABLE I
Parsers and options used

Parser	Possible options	Messages
`caradoc`	`extract`	121
	`stats`	121
	`stats --strict`	94
`hammer`	(none)	69
`mutool`	`clean`	214
	`draw`	248
	`show`	75
`origami`	`pdfcop`	40
`pdfium`	(none)	26
`pdfminer`	`dumppdf`	88
	`pdf2txt`	155
`pdftk`	`server`	33
`pdftools`	`pdfid`	4
	`pdfparser`	30
`peepdf`	(none)	4
`poppler`	`pdffonts`	100
	`pdfinfo`	90
	`pdftocairo`	214
	`pdftoppm`	155
	`pdftops`	189
	`pdftotext`	139
`qpdf`	(none)	192
`verapdf`	`greenfield`	40
	`pdfbox`	50
`xpdf`	`pdffonts`	82
	`pdfinfo`	70
	`pdftoppm`	122
	`pdftops`	157
	`pdftotext`	100
Total		3022

A. Message regex construction

The message regexes were generated by running all unique `stderr` messages for each parser (independently in parallel) through a set of manually created find-and-replace rules, followed by a final multi/single line filter, as described below. The idea is that message regexes should match to a message type or template, ignoring variable fields. For example, one of the caradoc stderr messages is `PDF error : Error in Flate/Zlib stream in object` *`[number]`*`!`, and messages which fill in the template with different object numbers should all still be combined into the same regex: `PDF error : Error in Flate/Zlib stream in object \d+ !`

Each rule includes (1) a `stderr`-file-wide regex, (2) a find regex, and (3) replace text. If the rule's `stderr`-file-wide regex matches the `stderr` file produced by a parser, then the rule's find regex is used to replace all *its* matches with the rule's replace text to create a potential message regex.

When a given file is run through a parser, the parser produces a `stderr` stream that often contains several messages. Our methodology requires the extraction of all messages from

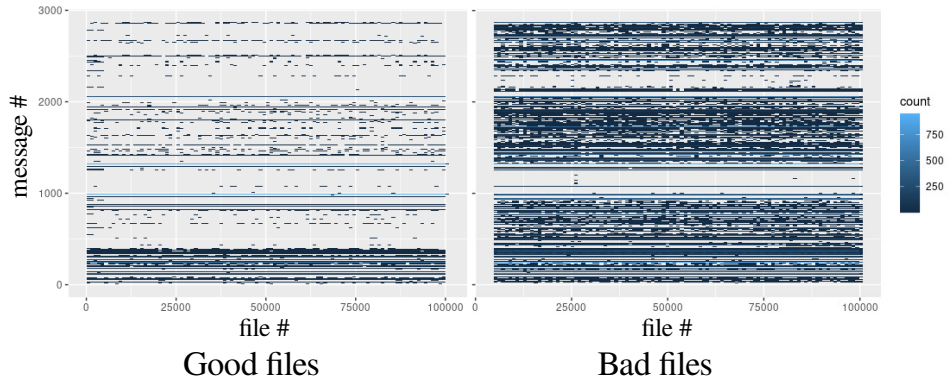


Fig. 1. Matrices of messages (rows) versus files (columns) for the SafeDocs Evaluation 3 Universe A good files (left) and bad files (right). The message numbers are arbitrary but are the same for both matrices. The files differ between the two data sets.

this `stderr` stream. Because a given message produced by a parser may span multiple lines of text, newlines cannot be automatically treated as a message delimiter when reading `stderr`. To manage this situation, our regex set contains contains duplicates of single and multi- line messages in different orders. This construction allows us to simply match each regex against the entirety of the `stderr` stream.

B. Preliminary test: Message independence

To assess the message conditional independence assumption, we ran χ^2 tests for independence on each pair of messages drawn from a simple random sample of $n = 30$ messages for the good files and the bad files, separately. Figure 2 shows a summary of the resulting p-values for each pair of messages from the sample for both the good (left) and bad (right) files. A small p-value corresponds to a likely dependent pair of messages, while a large p-value corresponds to a pair of messages that are likely independent. Figure 2 shows that most message pairs are likely independent (with $p > 0.05$, though typically much larger). There is a small subset of the message pairs that are very dependent (with $p < 0.05$). Given the banding structure, these dependencies are caused by a small number of individual messages.

Some of the most significant message dependencies correspond to duplicate regexes run on the output of a given parser, when different options are enabled. For instance, messages 19 and 140 in Table V use the same regex for the output of the `caradoc` parser. Message 19 is reported when `caradoc` is run with the `extract` option, while message 140 is reported when `caradoc` is run with the `stats` option. Given that running `caradoc` with different options likely uses the same code in many places, is reasonable to expect—though it need not be the case that—a given file will produce both of these messages or neither of them.

III. Methods

Let A and B be two sets of files, corresponding to different dialects that we would like to classify. That is, files in A are of one dialect, while files in B are of another dialect. Each of these files are run through parsers that can potentially produce any messages from a fixed set M of messages. We will assume that the messages are independent as random variables, after they have been conditioned upon the dialect. That is, if we are only considering files of dialect A, then the messages will be independent. However, if we consider the files in two dialects $A \cup B$, then independence may be violated.

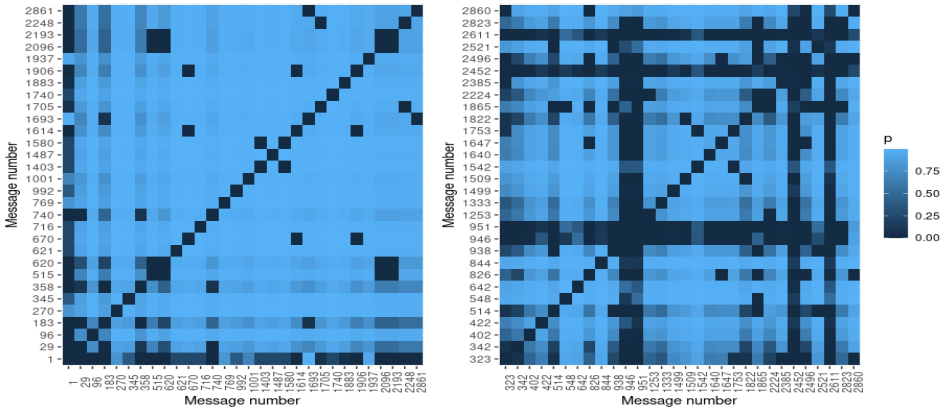
The independence assumption lets us consider the message probabilities for each dialect separately. In the case of dialect A, we will model the messages as Bernoulli random variables with one of two probabilities: p_0 or p_A. The subset of messages with probability p_A is denoted as $M_A \subseteq M$. The remainder of the messages (in $M_A^c = M - M_A$, the complement of M_A within M) are assumed to occur with probability p_0. If we assume that $p_A > p_0$, it is useful to interpret M_A as the set of messages that are "characteristic" to dialect A. We can similarly define a set M_B of messages that occur with probability p_B for files in dialect B.

We will use only one value for p_0 across both dialects, which suggests the interpretation that p_0 is the "background" message probability. If message is in $M_A \cap M_B^c$, then it will either occur with probability p_A if the file is in dialect A, or it will occur with probability p_0 if the file is in dialect B. Conversely, a message in $M_A^c \cap M_B$ will occur with probability p_B if the file is in dialect B, or it will occur with probability p_0 if the file is in dialect A.

Under the above assumptions, the probability of getting exactly a set of messages $K \subseteq M$ on a file f in dialect A is

$$\begin{aligned} P(K|A) = & p_0^{\#(K \cap M_A^c)}(1-p_0)^{\#(K^c \cap M_A^c)} \times \\ & p_A^{\#(K \cap M_A)}(1-p_A)^{\#(K^c \cap M_A)}. \end{aligned} \tag{1}$$

We note that in our two datasets, the messages are not completely independent, especially for those messages that have a low probability of occurrence. The proper probabilistic model should account for various correlations between messages, but is substantially more complicated than Equation (1). That said, by artificially inflating the probability p_0, one can produce similar message patterns to what are observed in the data.



Good files Bad files

Fig. 2. SafeDocs Evaluation 3 Universe A dataset pairwise χ^2 test for independence between a random sample of 30 messages. The p-values for each pair are shown for the good files (left) and bad files (right).

A. Estimating the parameters of the model for a single dialect

If $\#M_A \ll \#M$, then we can estimate p_0 from the data. The probability that no messages will occur is then

$$\begin{aligned} P(\emptyset|A) &= (1-p_0)^{\#M_A^c}(1-p_A)^{\#M_A} \\ &= (1-p_0)^{\#M}\left(\frac{1-p_A}{1-p_0}\right)^{\#M_A} \\ &\approx (1-p_0)^{\#M}. \end{aligned} \tag{2}$$

On the other hand if $p_A \gg p_0$ is a good assumption, then considering each message independently will identify those that are in M_A, since their individual probabilities differ from p_0 by a significant amount.

Although it turns out to be unnecessary in the case of our dataset, one could identify messages in M_A by a standard hypothesis test for a proportion. This is helpful if p_A is close to p_0. To that end, if the t-test statistic for message k,

$$t = \frac{p_k - p_0}{\sqrt{\frac{p_0(1-p_0)}{\#\text{files}}}}$$

is large—say larger than 1.96 for 95% confidence—then we conclude that this message is an element of M_A.

B. Message patterns and weighted Dowker complexes

Given the set of messages M, there are 2^M possible *message patterns* that might occur for a given a file. Under the model given by Equation (1), not all of these are equally likely. Some message patterns can be expected to be quite common. For instance, if the messages are all "errors" and the dialect under consideration consists of mostly valid files, we should expect the the empty pattern $\emptyset$ to be the most common.

The set of message patterns that occur for a given dataset has rich mathematical structure. The most famous of these structures is that of the *Dowker complex*.

Definition 1. [2], [16] The set X of all message patterns $K \subseteq 2^M$ such that there is a file exhibiting (at least) the messages in K is called the *Dowker complex*. Each such message pattern is called a *Dower simplex*. Furthermore, the number of files exhibiting *exactly* the messages in K *and no others* is called the *differential weight* $d(K)$. For simplicity, we will usually call $d(K)$ the *weight* or the *file count* for the message pattern K.

There are many interesting properties of the Dowker complex, because it is an example of an *abstract simplicial complex*, a combinatorial topological model of an abstract space. For our purposes in this article, the most important properties follow from the fact that message patterns are *partially ordered* by *subset inclusion*. That is, if K_1, K_2, and K_3 are message patterns and we know that $K_1 \subseteq K_2$ and $K_2 \subseteq K_3$, then it follows that $K_1 \subseteq K_3$. Moreover, if $K_1 \subseteq K_2$ and $K_2 \subseteq K_1$, then it follows that $K_1 = K_2$. It is obvious that the message count for a message pattern K (the number of messages in K) constrains which other message patterns are related to K. This impacts the statistics of the distribution of message counts exhibited within a given dataset, as will be explained in Lemma 1 of Section III-C.

Since the number of messages is typically large, for instance $\#M = 3022$, the number of possible message patterns is truly enormous. It is therefore unwise to attempt to compute the weight of all possible message patterns for a given dataset. Since most of these weights will be zero, it is much better to compute the message patterns that are actually present and their corresponding weights simultaneously. This can be done efficiently by lexically sorting the columns of the matrix form of the data, and then grouping blocks of identical columns greedily. Each distinct column clearly corresponds to a particular message pattern with a nonzero weight, and the weight is simply the number of duplicate columns.

As a sample implementation of this greedy approach, we exhibit a very succinct implementation in the `tidyverse` dialect of the R statistical programming language. (A more optimized Python implementation, suitable for interactive exploration of our entire test data, is discussed in Section IV-C.)

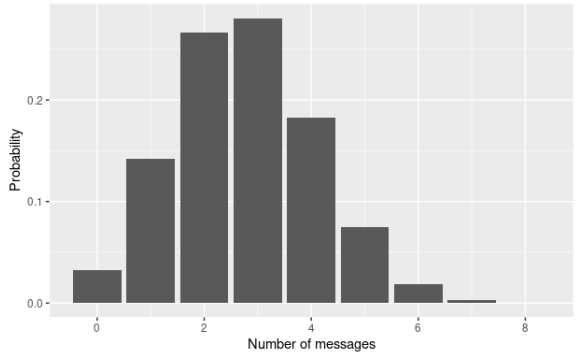


Fig. 3. Probability that a file will produce a certain number of messages, given a total of 8 messages: 3 messages with probability 0.25, and 5 messages with probability 0.4.

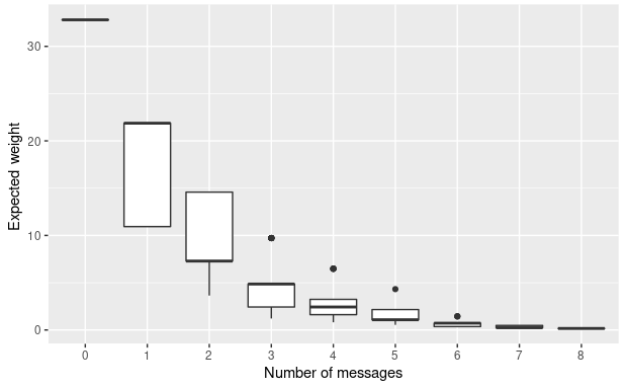


Fig. 4. Expected weight (file count) of a simplex compared with its message count, given a total of 8 messages: 3 messages with probability 0.25, and 5 messages with probability 0.4.

This implementation assumes that the Boolean matrix of messages is stored as a data frame called `message_data`, in which the rows correspond to files. (That is, the data frame is the transpose of the matrices shown in Figure 1.) The columns which correspond to messages are named starting with the character `'X'`, and all other columns are ignored.

The following snippet will compute a new data frame called `dowker` containing the Dowker complex and weights (file counts), and then produces a histogram of the weights, like what is shown in Figure 4. Because the `count()` function is quite efficient, the snippet runs quickly provided that the data fit in memory.

```
dowker <- message_data %>%
  group_by(across(starts_with('X'))) %>%
  count(name='weight',sort=TRUE) %>%
  ungroup()
dowker %>%
  mutate(simplex=row_number()) %>%
  ggplot(aes(x=simplex,
             y=sort(weight,
                    decreasing=TRUE))) +
  geom_line()
```

C. Relationship with message count

If the messages mostly correspond to error conditions and most files in a given dialect are valid, then we expect that most files will generate few messages. Using the model given by Equation (1), the probability that a file will produce n messages is the weighted sum of binomial distributions,

$$P(\#K = n|A) = \sum_{k=0}^{n} \left[\binom{\#M_A}{k} \binom{\#M - \#M_A}{n-k} p_0^{n-k} \times (1-p_0)^{\#M - \#M_A - (n-k)} \times p_A^k (1-p_A)^{\#M_A - k} \right]$$

After a bit of algebra (or logic), if $p_A = p_0$, then the probability of n messages occurring is simply given by a binomial distribution. Since the binomial distribution is not necessarily monotonic, neither is message count, for instance see Figure 3. Nevertheless, there is a definite dependence between message count and the probability of a given message pattern.

Lemma 1. *Assume that each message has probability less than* 0.5 *and the messages are independent when conditioned on files of dialect* A*. If* $K_1 \subset K_2$ *are two message sets, so that* $\#K_1 < \#K_2$*, then* $P(K_2|A) < P(K_1|A)$*.*

Proof. By moving from K_1 to K_2, we are merely swapping out factors in $P(K|A)$ of the form $(1-p_k)$ for corresponding factors p_k. Under the hypothesis, these new factors are smaller. □

If messages occur with probability greater than 0.5, it is usually more informative to consider their *absence* instead of their presence. For instance, `pdfium` usually produced a message `Processed \d+ pages\.`. The absence of this message suggests that the parser exited without producing any useful output because the file was too malformed, but its presence was not very helpful.

Note that the Lemma is clearly true for Equation (1), though it holds even if every message probability is different. Inspired by the binomial distribution, we expect the weight to eventually decrease as number of messages increases, as shown in Figure 4. The variability in weights in Figure 4 is a bit misleading, because as messages are added the weight must decrease.

Corollary 1. *Under the same conditions as Lemma 1, if* K_1 *and* K_2 *are two Dowker simplices, with* $K_1 \subset K_2$ *(so that* K_2 *consists of more messages happening), then the expected weights satisfy* $d(K_1) > d(K_2)$*.*

That is, patterns with a higher message count are typically exhibited by fewer files. Note that this does not mean that the weight decreases with increasing message count; only that it decreases *as additional messages are considered*. As a result, it is particularly interesting when this trend is not followed in

the actual data: the weight *increases* as additional messages are added. These violations of Corollary 1 typically occur when messages are strongly dependent upon one another, and are effectively measuring related phenomena.

D. Distribution of weights

Observe that for a given pattern of messages K for a single dialect A, Equation (1) gives the probability that a particular file will count towards the weight for K, when K is thought of as a simplex of the Dowker complex.

A convenient display of the weights for a given set of message patterns is to sort them in decreasing order, resulting in a *Dowker histogram*. Equation (1) specifies the expected values of each possible weight, but does not specify how many simplices will have this particular weight. This is easily found, however. There are

$$\binom{\#M_A}{k}\binom{\#M - \#M_A}{n-k}$$

different simplices corresponding to the situation where n messages occurred, k of which are characteristic to the dialect A. Each of these simplices has the same expected weight, namely

$$P(\#K = n, \#(K \cap M_A) = k|A) = p_0^{n-k}(1-p_0)^{(\#M - \#M_A - (n-k))} p_A^k (1-p_A)^{\#M_A - k}. \tag{3}$$

Example 1. Convergence of the actual weights to the expected weight computed by Equation (3) is quite rapid if p_0 and p_A are not close to 0.5. Consider a dataset with 8 messages collected from 1000 files, in which 5 of the messages occur with probability 0.4 and the remaining 3 messages occur with probability 0.25. Such a dataset is much smaller than the dataset we ultimately considered! To assess the variability in the resulting Dowker histogram, we simulated 300 cases of this kind of dataset. The resulting histograms are shown (in aggregate) in blue in Figure 5, over which the expected weights are plotted in red. There is quite close agreement between the simulated and expected histograms.

E. File ratios for two dialects

Suppose that the dataset contains files from two dialects, A and B. Although files from both dialects might exhibit a given message pattern, this pattern may occur more frequently for files from one dialect. Another way to interpret this situation is that files of one dialect may be more prevalent on certain Dowker simplices than on others. If the distribution of files across the simplices differs, then it is possible to separate (some portion of) the dialects. There can be simplices where the dialects overlap, namely certain patterns of messages that are exhibited with roughly equal probabilities by both dialects. These files cannot be separated using message patterns, and are of potential interest to file format analysts.

The ratio of dialect B files to dialect A files for a message pattern K is

$$\frac{P(K|B)(\#B \text{ files})}{P(K|A)(\#A \text{ files})} = \left(\frac{P(K|B)}{P(K|A)}\right)\left(\frac{P(B)}{P(A)}\right)(\#\text{files}).$$

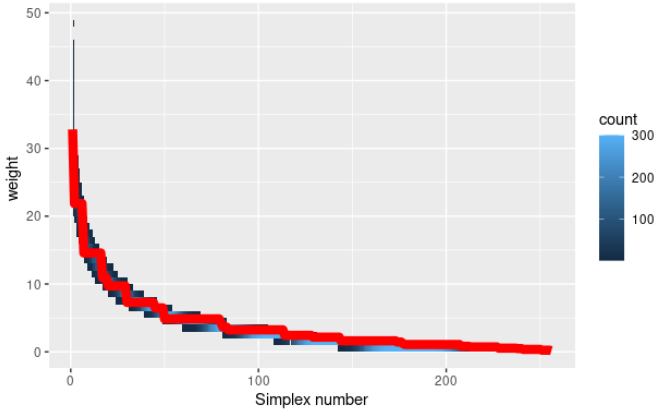


Fig. 5. Histogram of expected Dowker weights (red) versus 300 random trials (blue/gray) of 1000 files, with a total of 8 messages: 3 messages with probability 0.25, and 5 messages with probability 0.4. Each point along the horizontal axis specifies a simplex (ordered so that their corresponding weights decrease).

Notice that the ratio of conditional probabilities is the only dependence on the simplex K. This ratio can be estimated using Equation (1).

Lemma 2. *If $M_A \cap M_B = \emptyset$, then the ratio of conditional probabilities for B files to A files in simplex K is expected to be*

$$\frac{P(K|B)}{P(K|A)} = \left[\left(\frac{p_0}{p_A}\right)\left(\frac{1-p_A}{1-p_0}\right)\right]^{\#(K\cap M_A)} \times \left[\left(\frac{p_B}{p_0}\right)\left(\frac{1-p_0}{1-p_B}\right)\right]^{\#(K\cap M_B)} \times \left(\frac{1-p_0}{1-p_A}\right)^{\#M_A}\left(\frac{1-p_B}{1-p_0}\right)^{\#M_B}.$$

If the conditional independence assumption is violated, the ratio of conditional probabilities can still be computed. In this case, it is usually called a *pseudolikelihood ratio* [17].

Proof. This is an elaborate calculation following from Equation (1), the main goal of which is to eliminate the complements where they appear on each of K, M_A and M_B. The following Boolean algebraic identities help to simplify the work:

$$\begin{aligned} K \cap M_A^c &= (K \cap M_A^c \cap M_B^c) \cup (K \cap M_A^c \cap M_B) \\ K \cap M_B^c &= (K \cap M_A^c \cap M_B^c) \cup (K \cap M_A \cap M_B^c) \\ K^c \cap M_A^c &= (K^c \cap M_A^c \cap M_B^c) \cup (K^c \cap M_A^c \cap M_B) \\ K^c \cap M_B^c &= (K^c \cap M_A^c \cap M_B^c) \cup (K^c \cap M_A \cap M_B^c) \end{aligned}$$

Specifically, the first two identities, followed by an application of the disjointness $M_A \cap M_B = \emptyset$, establishes that

$$\frac{p_0^{\#(K\cap M_B^c)}}{p_0^{\#(K\cap M_A^c)}} = p_0^{\#(K\cap M_A\cap M_B^c) - \#(K\cap M_A^c\cap M_B)} = p_0^{\#(K\cap M_A) - \#(K\cap M_B)}.$$

In a similar way, we can derive that

$$\frac{(1-p_0)^{\#(K^c\cap M_B^c)}}{(1-p_0)^{\#(K^c\cap M_A^c)}} = (1-p_0)^{\#(K^c\cap M_A\cap M_B^c)-\#(K^c\cap M_A^c\cap M_B)} = (1-p_0)^{\#(K^c\cap M_A)-\#(K^c\cap M_B)}.$$

Reorganizing yields the ratio

$$\frac{P(K|B)}{P(K|A)} = \left(\frac{p_0}{p_A}\right)^{\#(K\cap M_A)}\left(\frac{p_B}{p_0}\right)^{\#(K\cap M_B)}\times \left(\frac{(1-p_0)}{(1-p_A)}\right)^{\#(K^c\cap M_A)}\left(\frac{(1-p_B)}{(1-p_0)}\right)^{\#(K^c\cap M_B)}.$$

To remove the remaining complements on the K, observe that

$$\#(K^c\cap M_A) = \#M_A - \#(K\cap M_A)$$
$$\#(K^c\cap M_B) = \#M_B - \#(K\cap M_B)$$

from which the desired result follows. □

Under the assumption that both p_A and p_B are greater than p_0, the first factor in the statement of Lemma 2 is less than 1, while the second is greater than 1. Therefore, the ratio of dialect B to dialect A files in the simplex corresponding to K is increased by ensuring that the messages in K contain all of M_B and none of M_A. Messages outside $M_A \cup M_B$ do not impact the ratio of files in K at all. This is sensible: if one wishes to collect mostly dialect B files, one looks for those that produce any messages in M_B but none in M_A.

Conversely, places where the file ratio is close to 1 are message patterns that are ambiguous. Format analysts should spend more time on that particular set of files, since it is hard to disposition the files as clearly one dialect or the other without inspecting the file contents directly. One could imagine that it might also be possible to craft new messages that discriminate between the dialects better by considering *only* the files exhibiting these ambiguous message patterns. This may greatly reduce the number of files that need to be considered.

Corollary 2. *If $M_A \cap M_B = \emptyset$ and $p = p_A = p_B$, then the ratio of B files to A files in simplex K is expected to be*

$$\frac{P(K|B)}{P(K|A)} = \left[\left(\frac{p}{p_0}\right)\left(\frac{1-p_0}{1-p}\right)\right]^{\#(K\cap M_B)-\#(K\cap M_A)}\times \left(\frac{1-p}{1-p_0}\right)^{\#M_B-\#M_A}.$$

Corollary 2 is straightforward to apply and gives fine-grained information about where to look for files of a certain dialect.

Example 2. Consider a notional dataset containing files from two dialects A and B. Suppose that there are three messages in total with $\#M_A = \#M_B = 1$, and $M_A \cap M_B = \emptyset$. If we let $p = p_A = p_B = 0.4$, $p_0 = 0.2$, this dataset will satisfy the hypotheses of Corollary 2.

For three messages, there are 8 possible message patterns. These message patterns can be organized in a lattice based upon incrementally adding new messages, as is shown in Figure 6. In the figure, the message patterns are indicated by three squares: a filled square indicates that the corresponding message occurred, while an empty one indicates that the message is absent. Message count increases as one moves up in the diagram: the bottom row corresponds to files exhibiting no messages, while the top row corresponds to files exhibiting all three messages.

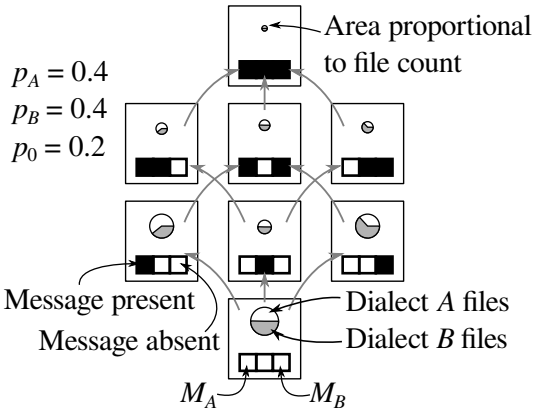


Fig. 6. Expected weight and file ratio for two dialects using three messages as described in Example 2.

Corollary 1 asserts that because $p, p_0 < 0.5$, the per-dialect weights (shown as areas in the pie charts in Figure 6) must decrease as one follows the gray arrows upwards in the diagram (adding messages incrementally). Corollary 2 was used to compute the expected file ratio for each message pattern.

The largest weight is on the bottom of the diagram, where the files exhibit no messages. Since the ratio is 1 for that message pattern (the pie chart shows equal areas for both dialects), it is not possible to separate dialects for the files that exhibit no messages. One might imagine that there are other features that might be informative, but that these are simply not captured by the three messages under consideration.

The next largest weight is in the second row, where just the messages in M_A or M_B occur. (If $p > 0.5$ instead, the largest weights can occur on rows other than the bottom, because the hypotheses of Corollary 1 are not satisfied.)

Considering the pie charts, the majority of the dialect A files are on the left of the diagram, while the majority of the dialect B files are on the right of the diagram. Therefore, coarse dialect separation can be done on those files on the left or right, on account of their message patterns being different.

F. Separating dialects by thresholding posterior probabilities

Of course, the disjointness of M_A and M_B required by Lemma 2 might not hold exactly in practice. Messages that lie in the intersection of M_A and M_B will not impact the ratio of files on any simplex if $p_A \approx p_B$, as is also easily seen in Corollary 2. However, if p_A and p_B differ substantially, this will tend to change the ratio of files from the estimate

in Lemma 2 for any simplex that contains any messages in $M_A \cap M_B$. Furthermore, if there are not many B files, it may be difficult to estimate p_B or the true contents of M_B.

Observe that files in the tail of the Dowker histogram are also in the tail of the message count histogram. These are instances where $P(K|A)$ or $P(\#K = n|A)$ is low. On the other hand, if $P(K|B)$ or $P(\#K = n|A)$ is comparatively higher, this will cause the ratio of dialect B files to be statistically significantly higher, and thereby possible to detect.

A systematic way to exploit this information is to consider the probability that a given file is from a certain dialect, given that it exhibits a certain message pattern. This is called the *posterior probability*, and can be computed using Bayes' theorem. For instance, to ascertain the probability that a file is in dialect A given that it produced message pattern K, this is given by

$$P(A|K) = P(K|A)\frac{P(A)}{P(K)}. \tag{4}$$

This formula defines a test statistic—a quantity that yields a classifier for dialect A files upon thresholding $P(A|K)$. Files with large $P(A|K)$ are likely from dialect A, while files with a lower value of $P(A|K)$ are less likely to be from dialect A.

In order to compute $P(A|K)$, one needs to obtain each factor in Equation (4). $P(K)$ is easily assessed by computing the frequency of the message pattern K in the dataset at hand. $P(K|A)$ is given by Equation (1), subject to the assumptions mentioned earlier in the paper. Moreover, if one has a training set consisting (almost) entirely of dialect A files, then one can estimate $P(K|A)$ by simple counting. This avoids the potential issues with violations of the independence of messages.

Finally, $P(A)$ is the expected probability that a file will be of dialect A, given no further information. This last factor is the most difficult to estimate, and can be best thought of as a "risk factor": choosing a larger value of $P(A)$ will result in a higher estimate for $P(A|K)$, while choosing a lower value of $P(A)$ will consequently reduce the estimate for $P(A|K)$. Thus, if dialect A files are dangerous, it is wise to overestimate $P(A)$. Conversely, if A files are likely benign, an underestimate of $P(A)$ will produce fewer false alarms.

IV. Results

Figure 7 shows the message probabilities for the SafeDocs Evaluation 3 Universe A good files dataset. Some messages occur more than 50% of the time; for these messages, it is more useful to assume that their *absence* is an informative event. Therefore, as a preprocessing step, when computing the probability of a message k, if its probability is $p_k > 0.5$, we instead use $1 - p_k$ in what follows.

The vast majority of messages have a low probability of occurrence, while there are roughly 6 messages with an elevated probability (note the logarithmic scale). Using the model we propose is tantamount to discretizing the probabilities shown to two separate levels: a higher one for the first 6 messages, and a lower one for the rest of the messages. The average probability for each of the first 6 messages is $p_{\text{good}} = 0.380$. The specific messages this threshold selects are shown in Table II, so a reasonable cutoff for the probability of an M_{good} message is $p_{\text{good}} > 0.25$. We therefore take this as a definition of the M_{good} messages in what follows. (Note that Table II shows the raw probabilities of the messages, which may exceed 0.5.) The regexes for these messages appear in Table V. It happens that nearly all of the M_{good} messages are nonzero exit codes for parsers, but with no further detail. Four of the messages are from `caradoc`, which is known to be a fairly stringent parser. One may interpret M_{good} as consisting of mostly benign messages that are indicative of otherwise good files.

TABLE II
Messages in M_{good} for Universe A files

Message	Parser and options	Prob. in good files	Prob. in bad files
1	`caradoc extract`	0.414	0.697
122	`caradoc stats`	0.414	0.697
943	`origami pdfcop`	0.426	0.500
2055	`qpdf`	0.303	0.603
243	`caradoc stats --strict`	0.626	0.842
334	`caradoc stats --strict`	0.351	0.033

Running the same process on the SafeDocs Evaluation 3 Universe A bad files dataset with same threshold as before, $p_{\text{bad}} > 0.25$, yields 54 messages with an elevated probability, at approximately 0.312. The intersection between this set and the corresponding set of messages for the good files is nonempty and consists of 4 messages, shown in the upper portion of Table II. The actual regexes appear in Table V.

When there are many messages, the estimates of $P(K|A)$ tend to be very small, regardless of the dialect, and therefore are subject to substantial sampling error. In the Universe A good files set, none of the files produced no messages (after inverting the meaning of any message with probability greater than 0.5). Therefore Equation (2) cannot be used. Moreover, many of the messages with low probabilities are not independent. For these reasons, it is better to select an overestimate for p_0, since this results in more frequent co-occurrence between messages. The largest overestimate for p_0 is the value at the threshold chosen for M_{good}, namely $p_0 = 0.25$, as shown on Figure 7. This choice will be later confirmed by the agreement between the actual and expected Dowker histograms described in the next Section by considering Figure 8.

A. Message pattern distribution

There are $2^{\#M}$ possible message patterns. Since the total number of messages collected was large, it is not feasible to compute the expected weights for each message pattern. Instead, it is much more practical to compare only the message patterns that were actually observed. This means that we need to normalize the probabilities so that the sum over all *observed* message patterns is 1, instead of normalizing so that the sum over *all possible* message patterns is 1. This being done, the comparison between observed and expected weight distributions is shown at left in Figure 8. There is close agreement over the entire distribution, which can be taken as

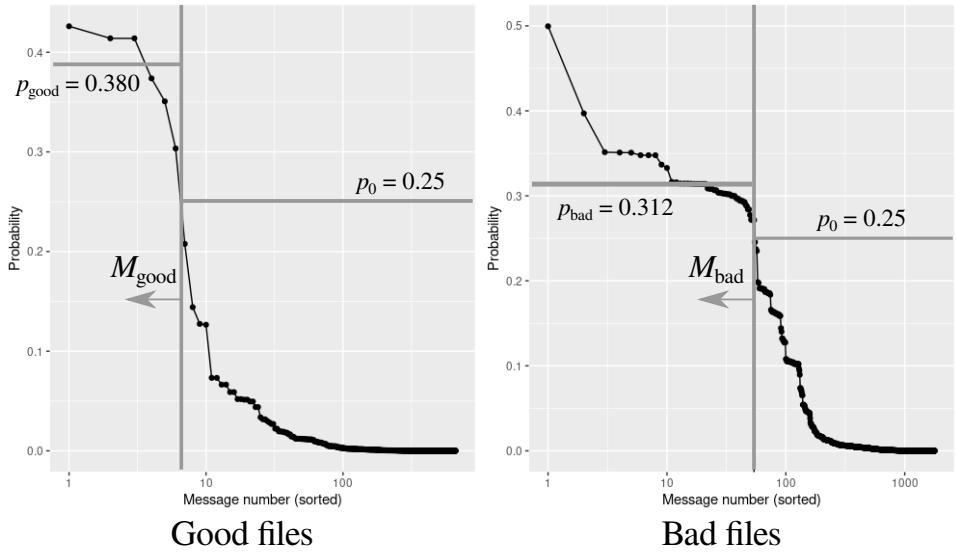


Fig. 7. Probability a given message will occur for the SafeDocs Evaluation 3 Universe A good files (left) and bad files (right). In both frames, messages in M_A are to the left of the gray vertical line. Estimates for p_A and p_0 are shown as horizontal lines.

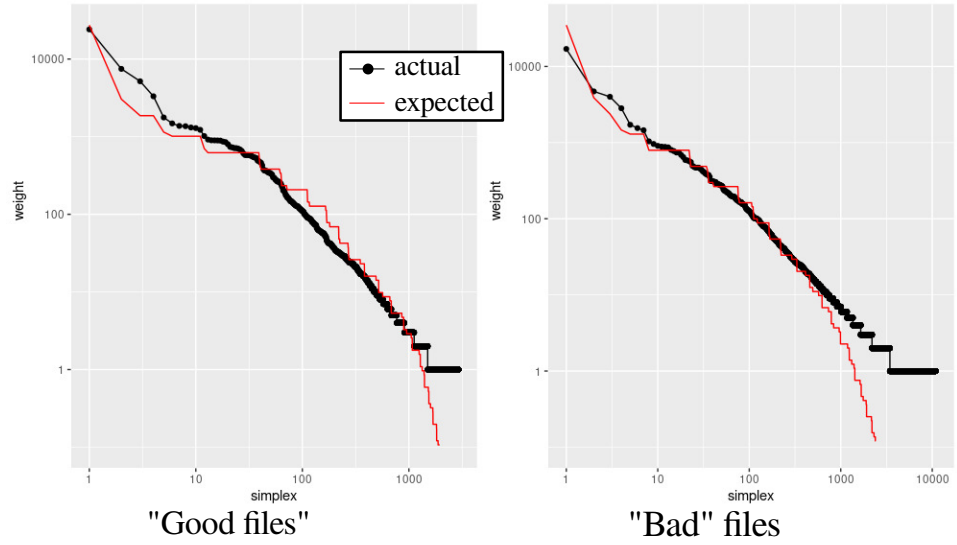


Fig. 8. Histograms of expected weights (file counts) versus actual weights for the SafeDocs Evaluation 3 Universe A datasets: (left) the good file subset, used for training, and (right) the bad files subset.

validating our model of the data, and validating our choice of p_0 in particular.

We can also compare the expected distribution of weights with the bad files set. It is most interesting to do this not with M_{bad} and p_{bad}, but rather with the parameters M_{good}, p_{good}, and p_0. Because of the close agreement on the good files, differences between the expected and observed distributions in the bad files are more visible. The resulting plot is shown at right in Figure 8. While there is still close agreement for the large weights (indeed, there is somewhat better agreement than for the good files), the message patterns with low weights are quite different. The expectation is that there would be rather fewer files exhibiting particular message patterns than actually occurred. Since low weights are correlated with higher message counts, this suggests that the bad files frequently produce messages that are not in M_{good}. In other words, the bad files consist of a distinct dialect from the good files.

B. Behavior differences within the datasets

Grouping files based upon message patterns does yield a useful tool for clustering related behaviors. This is exhibited in both the good and bad files, with the largest weights corresponding to clear behavioral patterns.

The message patterns with the largest weights are shown in Table III. Since messages 243, 991, 1319, and 287 had probabilities exceeding 0.5, we will consider their absence

rather than their presence as noted earlier. For instance, `pdfium` always produces message 991, so it is rather uninformative. The messages that were initially reported (without this reversal) are shown in the column "Raw messages", while the messages after this reversal are recorded in the column "Messages (corrected)".

Although these messages are recoverable (since the files under consideration are in the good files set), these message patterns correspond to three distinct classes of behavior: errors within a compressed stream, syntax errors within the PDF file, and type errors.

The specific messages involved in Table III are shown in Table V. Several of the regexes are duplicated because the same base parser `caradoc` can be run with different options. (These duplicated messages are not independent.)

For the bad files, the message patterns with the largest weights involve substantially more messages, as shown in Table IV. Again, several messages occur with probability greater than 0.5 and so their absence is shown in the column "Messages (corrected)". These message are different from the good files, and are 1, 122, 243, 991, 1319, 2055, and 2287. Without this correction, the most common message pattern in the bad files is the same as one occurring in the good files, involving a syntax (lexing) error. However, without the correction, it is rather different. This message pattern contains $5170/(5170 + 16980) = 23\%$ good files and 77% bad files. Further delineation of this particular syntax error as recoverable or not is impossible given the messages we collected.

Aside from the most common pattern, there is no overlap between good and bad files among the most common message patterns. In the remaining message patterns most commonly exhibited by the bad files, there are several patterns corresponding to damaged `xref` tables. Since `xref` tables provide important structural information about the contents a PDF file, it is unsurprising that damage to the `xref` table results in many more messages being produced.

From a careful inspection of the first two rows of Table III, we can conclude that there is a violation of Corollary 1. That is, the addition of message 943 resulted in a higher weight with than without it. This indicates that there is a strong relationship between message 943 and the messages which indicate issues with compressed streams. We can infer that the presence of message 943 is sometimes indicative of problems with compressed streams in PDFs, even though the message is merely an exit code.

A violation of Corollary 1 was also exhibited by the bad files as well, though it does not appear in Table IV because the lists of messages are too long to fit. This violation effectively groups together a collection of messages related to broken `xref` tables.

C. Interactive display of Dowker complexes

While the Dowker complex can be computed succinctly using R as discussed in Section III-B, this implementation is not particularly efficient for large datasets. Additionally, it does not easily support visualization of the lattice structure mentioned in previous sections. Therefore, we developed an optimized Python version of the Dowker complex construction. For a small number of messages, a 2d representation of the lattice structure (like what appears in Figure 6) suffices, but this becomes increasingly cluttered with more messages.

To remedy this issue, we implemented a Python version that embeds the Dowker complex in 3 dimensions and permits interactive examination. The Dowker visualization relies on `Plotly` and its 3d network graph [18]. The 3d network graph consists of connected nodes. Each node corresponds to a message pattern whose weight exceeds a user-chosen parameter, and each edge corresponds to the addition of a single message. The resulting graph can be customized by setting node and edge colors, positions etc.

We start with the Boolean matrix representation of the data stored in an array `msgMatrix` in which each file is a row and each message is a column (like in the R implementation, this is the transpose of the matrices shown in Figure 1). Each row corresponds to a message pattern, which will be displayed as a node in the graph upon the removal of duplicates. The first step is to construct the mappings of nodes to attributes as well as getting all possible connected nodes to be used to identify edges.

For efficiency, our implementation uses Python's `hash()` function to quickly and uniquely identify each message pattern. Importantly, this allows us to define a function `getConnNodes()` that takes a message pattern and computes the hashes for all possible message patterns with one fewer message. Using this function `getConnNodes()`, the pseudocode below shows how to construct the Dowker complex and its weights.

```
for row in msgMatrix:
 rowHash = hash(str(row))
 label = str(row)
 if rowHash in nodeWeightMap:
  nodeWeightMap[rowHash]+=1
 else:
  nodeWeightMap[rowHash]=1
  nodeLabelMap[rowHash] = label

  # Find labels for all possible
  # connected nodes, by finding
  # all nodes with 1 less message
  nodeConnNodeMap[label] =
    getConnNodes(row)
```

The above code is only notional because we found that the call to `hash(str(row))` turned out to be about 825 times slower than using `numpy` builtins to first interpret the row as bytes and then hexify into a string. This is because using `str()` on an array row calls `numpy`'s `array2str` with internal recursion that incurs about 100 operations per row and 20 operations per element while `numpy.packbits()` only incurs about 6 operations per row and 0 operations per element. Combined with other optimization efforts, the

TABLE III
THE MESSAGE PATTERNS WITH THE LARGEST WEIGHTS (FILE COUNTS) IN GOOD FILES

Weight	Raw messages	Messages (corrected)	Message count	Error taxonomy
24101	334, 943, 991, 1319, 2287	243, 943, 334	3	Compressed stream error
7470	334, 991, 1319, 2287	243, 334	2	Compressed stream error
5170	243, 258, 991, 1319, 2287	258	1	Syntax error (lexing)
3313	351, 991, 1319, 2287	243, 351	2	Syntax error (newline placement)
1767	1, 19, 122, 140, 243, 330, 991, 1319, 2287	1, 19, 122, 140, 330	5	Type error

TABLE IV
THE MESSAGE PATTERNS WITH THE LARGEST WEIGHTS (FILE COUNTS) IN BAD FILES

Weight	Raw messages	Messages (corrected)	Message count	Error taxonomy
16980	243, 258, 991, 1319, 2287	1, 122, 258, 2055	4	Syntax error (lexing)
4702	(not listed for space considerations)	(not listed for space considerations)	104	Damaged or missing `xref` table
4000	(not listed for space considerations)	(not listed for space considerations)	84	Damaged or missing `xref` table
2825	(not listed for space considerations)	(not listed for space considerations)	72	Damaged or missing `xref` table
1715	243,271,991,1319,2055,2287	1,122,271	3	Syntax error

TABLE V
MESSAGES INVOLVED IN TABLES II, III, AND IV

Message	Parser	`stderr` regex
1	`caradoc extract`	(exit code indicating error)
19	`caradoc extract`	`Type error : Unexpected entry .* in instance of class .* in object .* !`
122	`caradoc stats`	(exit code indicating error)
140	`caradoc stats`	`Type error : Unexpected entry .* in instance of class .* in object .* !`
243	`caradoc stats --strict`	(exit code indicating error)
258	`caradoc stats --strict`	`PDF error : Lexing error : unexpected character : 0x[A-Fa-f\d]+ at offset \d+ \[0x[A-Fa-f\d]+\] in file !`
271	`caradoc stats --strict`	`PDF error : Syntax error at offset \d+ \[0x[A-Fa-f\d]+\] in file !`
330	`caradoc stats --strict`	`Type error : Unexpected entry .* in instance of class .* in object .* !`
334	`caradoc stats --strict`	`Warning : FlateZlib stream with appended newline in object .*`
351	`hammer`	`VIOLATION\[\d+\]@\d+ \(0x[A-Fa-f\d]+\): No newline before 'endstream' \(severity\=.*\)`
943	`origami pdfcop`	(exit code indicating error)
991	`pdfium`	`Processed \d+ pages\.`
1319	`peepdf`	(exit code indicating error)
2055	`qpdf`	(exit code indicating error)
2287	`verapdf pdfbox`	(exit code indicating error)

Dowker generation was sped up by a factor of 628 over the naïve translation of the pseudocode.

Our preferred layout is a layered one, where each layer consists of all nodes with the same number of messages, and each layer is arranged in a circle. The layers are sorted numerically by message count. Other visualization methods we have tried include laying out nodes in a force-directed Kamada-Kawai and Fruchterman-Reingold [19], though the renderings these produced were generally harder to interpret because they disrupted the layered structure.

Figure 9 shows the Dowker graph generated from the SafeDocs Universe A combined dataset, with nodes colored such that higher weight nodes (more files triggering those message patterns) are colored yellow, and lower weight nodes are colored purple. The majority of files have few messages, as indicated by the wide "base" at the bottom of the rendering, where the brighter colored nodes are located. Nodes become more sparse at higher layers, corresponding to files that produced more messages. The wider "neck" in the the middle indicates that many message patterns had a message count around 100. Figure 9 also shows edges connecting neighboring message patterns which differ by a single message. The edges are colored in either green to indicate that the weight decreased in accordance with Corollary 1, or red if the weight increased. Because Corollary 1 depends on the conditional independence of messages, red edges indicate that this assumption has failed for the message patterns involved. Highly dependent messages are often indicative of dialect boundaries, so they could be candidates for further analysis based upon file contents. Moreover, the sparsity of edges in the upper portion of the diagram indicates that most of the message patterns for the associated files files are unrelated to one another; adding or removing a single message drastically reduces the number of files exhibiting that new pattern.

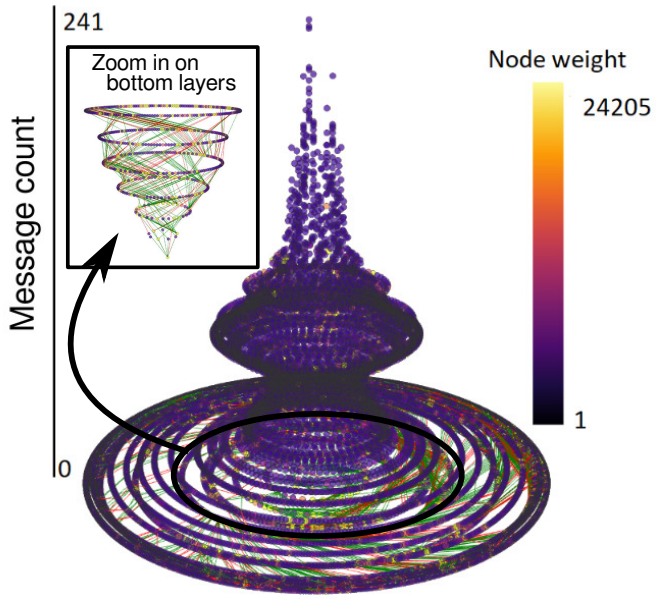


Fig. 9. 3d rendering of the Dowker complex for the SafeDocs Evaluation 3 Universe A, colored by weight. Each point corresponds to a message pattern, with the number of messages increasing as one moves up in the diagram. Edges marked in red correspond to violations of Corollary 1. Inset shows the ten "layers" corresponding to fewer than 10 messages.

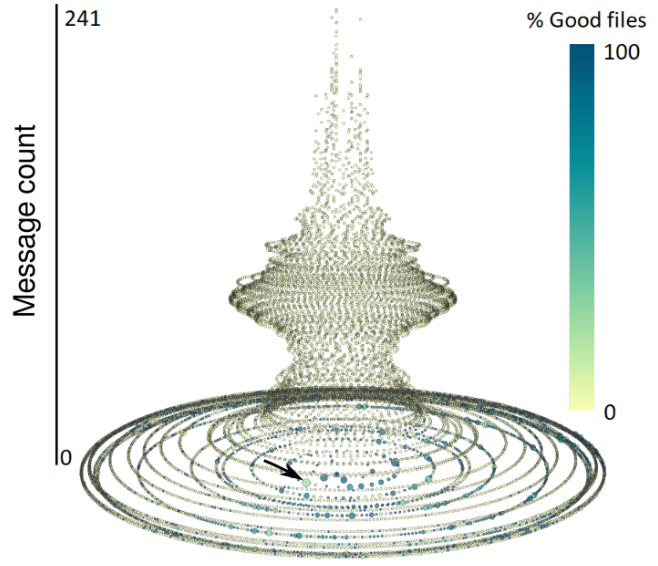


Fig. 10. 3d rendering of the Dowker complex for the SafeDocs Evaluation 3 Universe A, colored by percentage of good files. Each point corresponds to a message pattern. The size of the points indicates the weight of each message pattern. To reduce clutter, the edges are not shown.

We can also change the coloring of nodes to display the different dialects. Figure 10 shows the nodes' classification based on whether the files in question were good (blue), bad (yellow), or a mixture (shades of green). It is clear that the message count is clearly indicative of whether a file is good or bad: the nodes with high message counts are all yellow, corresponding to an overwhelming majority of bad files. Additionally, since the weights in Figure 10 are shown by the size of the nodes, the message patterns with the largest weights shown in Table III for the good files are quite visible, and they all appear near the bottom of the diagram. The largest weight message pattern for the bad files in Table IV is also visible near the bottom of the diagram as well, and is marked with an arrow. The presence of this particular message pattern, and several other majority-bad nodes with low message count near the bottom of the diagram justifies the use of the Dowker complex for dialect classification, rather than using only message count.

D. Separating dialects by thresholding

Thresholding posterior probabilities works well for separating the good files from the bad files. Starting with the Universe A good files as a training set ensures that we have a training set that is mostly good files.

From this training set, we estimate $P(K|\text{good})$ for all message patterns K that are exhibited in the data. To this end, we can either use Equation (1) (theoretical) based upon our previous estimates of M_{good} and $p_{\text{good}} = 0.380$, or we can compute $P(K|\text{good})$ directly by counting how many message patterns are exhibited (empirical).

Now let us consider the combined dataset with two dialects, namely both the Universe A good and bad files, but let us "forget" which file comes from which set. Given the fact that we know how many files of each dialect there are (but not which file is which), we know that $P(\text{good}) = 0.5$, since the data happen to contain equal numbers of both files. For this combined dataset, we can estimate $P(K)$ directly from the data by counting the number of times each message pattern occurs (just as we did for $P(K|\text{good})$).

Given all of these facts, we can then use Equation (4) to determine the probability $P(\text{good}|K)$ that a given file is good, given the particular message pattern K produced by the file. It still remains to select a probability threshold to use to determine whether we declare a file as good or bad. For that threshold, we can determine the *recall* (the fraction of good files with probability above our chosen threshold), and the *precision* (the fraction of truly good files above our threshold versus the total number of files above our threshold). An ideal classifier will have both precision and recall as close to 1 as possible.

Since we do not have any prior knowledge, the best measure of performance is to consider all possible thresholds, and to aggregate all precision and recall scores. This is shown in Figure 11. The figure shows three curves: the red curve uses Equation (1) to estimate $P(K|\text{good})$, the black curve uses the values estimated empirically from the good files alone, and the

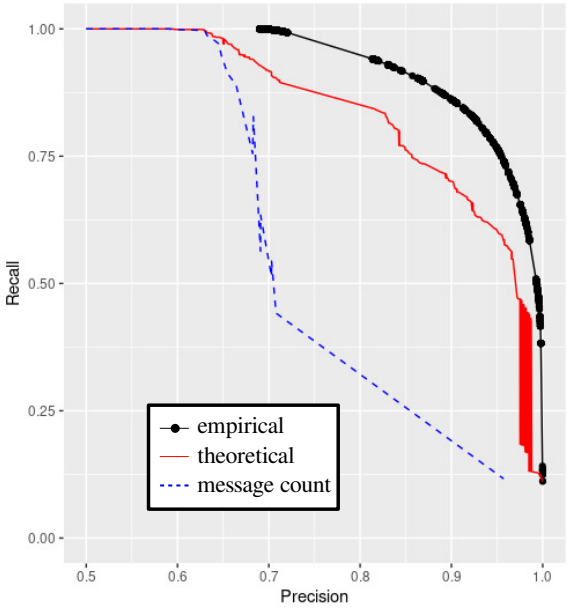


Fig. 11. Precision versus recall for separating the SafeDocs Evaluation 3 Universe A good files from the bad files, estimating $P(K|\text{good})$ empirically from training on the good files (black), using Equation (1) with parameters as described in the text (red), or using message count alone (blue, dashed).

blue dashed curve shows the performance of classifying using message count alone. While message count alone does not do a good job of classifying (largely because it misclassifies the bad files that produce only a few messages), the other two curves show good performance. Using the empirical estimates yields the best overall performance. This is not unexpected, since the conditional independence assumption made by Equation (1) does not entirely hold.

From Figure 11, we conclude that the precision of our method typically exceeds its recall for most threshold choices. One can interpret this to mean that the files with a high probability of being good based upon their message patterns are indeed good, though many good files are missed because they exhibit more unusual message patterns. Intuitively, this means that many messages produced are of a benign nature.

V. Conclusion

This paper provides a theoretical basis and practical algorithms for determining the format dialect of files within a dataset based upon the statistics of messages that parsers produce as they consume the files. The methods we used are based upon thresholding the posterior probability of a file being in a certain dialect, using the idea that messages occur independently once they are conditioned upon dialect. Even though a naïve classification of files based on message count might seem clearly the best, our method outperforms this by a wide margin.

Moreover, using our method, a format analyst can therefore greatly reduce the number of files they need to consider, by focusing their attention on *only* the files exhibiting message patterns with an ambiguous posterior probability. By looking at only these files, one can likely discover features that serve as "cut points" between dialects. Moreover, the theoretical file ratios allow one to predict which message patterns will be easy to disposition and which will not. Those that are not easy to disposition will tell the format analyst about what kind of new messages need to be crafted to discriminate between dialects. Such new messages are likely easier to construct under the condition that *the ambiguous message pattern is already present*. This may greatly reduce the number of files that need to be considered when constructing new messages.

Besides dispositioning of files as one dialect or another, the relationships between the message patterns themselves allow for a finer analysis. Our theoretical model establishes that the number of files exhibiting a given pattern of messages should decrease as more messages are triggered (Corollary 1). Violations of this result indicates places where our assumption of conditional independence is violated. In our data, these violations allow one to draw inferences about the semantic meaning of certain parser exit codes that are not associated with human-readable regular expressions.

Conversely, it is sometimes a valuable exercise to craft intentionally ambiguous files, which comply with multiple format specifications simultaneously. For instance, the *entire* journal issue [20] is a valid PDF file, ZIP file, JPEG file, and is—with a little work—compliant with several other formats! These so called *polyglot files* can be used to probe a format specification, as they often trigger unexpected corner cases in its logic. Intuitively, polyglot files are easiest to construct when they elicit a pattern of messages with file ratio close to 1. Knowing which message patterns already have file ratios close to 1 may aid in constructing these files.

The authors are presently collaborating with a software vendor to integrate the Dowker thresholding and visualization methodology into a PDF exploration tool. This tool allows for a user to easily query collections of files based on various properties, including the weight of various message patterns. It also supports visualizations such as those shown in Figures 9 and 10. In addition to studying message patterns, one can also study messages through the lens of *file patterns*. Although it has been shown by our team [16] that this is in some sense equivalent to what we present here, there are subtle relationships that are yet to be fully described. We are actively studying whether one can derive aspects of the meanings of the messages from the pattern of files that elicit them.

Given the simplicity of the Dowker histogram, as demonstrated by the code snippet in Section III-B, it is easy to apply the Dowker complex weighting methodology to many other settings. The generality of the Dowker construction means that it can be applied to any dataset containing two columns of categorical (names) data, not just file and message data. While the construction is mathematically well-defined, the underlying statistical models in other settings may differ from the conditional independence model presented here. Fortunately, even if there is no conditional independence, reasoning about the weighted Dowker complex is likely to

be statistically meaningful. It remains to be discovered how arbitrary statistical distributions are represented in the pattern of weights on the Dowker complex.

ACKNOWLEDGMENTS

The authors would like to thank the SafeDocs test and evaluation team, including NASA (National Aeronautics and Space Administration) Jet Propulsion Laboratory, California Institute of Technology and the PDF Association, Inc., for providing the test data. The authors would like to thank Denley Lam for the initial processing of the files into sets of messages.

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) SafeDocs program under contract HR001119C0072. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

REFERENCES

[1] M. Robinson, "Looking for non-compliant documents using error messages from multiple parsers," in *LangSec 2021, a subconference of IEEE Security & Privacy*, May 2021.

[2] K. Ambrose, S. Huntsman, M. Robinson, and M. Yutin, "Topological differential testing," 2020. [Online]. Available: https://arxiv.org/abs/2003.00976

[3] M. Belaoued and S. Mazouzi, "A real-time PE-malware detection system based on chi-square test and PE-file features," in *IFIP International Conference on Computer Science and its Applications*. Springer, 2015, pp. 416–425.

[4] B. A. S. Al-rimy, M. A. Maarof, and S. Z. M. Shaid, "Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions," *Computers & Security*, vol. 74, pp. 144–166, 2018.

[5] S. D. S.L and J. CD, "Windows malware detector using convolutional neural network based on visualization images," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, 2019.

[6] M. Alazab, "Profiling and classifying the behavior of malicious codes," *Journal of Systems and Software*, vol. 100, pp. 91 – 102, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121214002283

[7] J. Demme, M. Maycock, J. Schmitz, A. Tang, A. Waksman, S. Sethumadhavan, and S. Stolfo, "On the feasibility of online malware detection with performance counters," *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3, pp. 559–570, 2013.

[8] G. J. J. van den Burg, A. Nazábal, and C. Sutton, "Wrangling messy CSV files by detecting row and type patterns," *CoRR*, vol. abs/1811.11242, 2018. [Online]. Available: http://arxiv.org/abs/1811.11242

[9] K. Fisher, D. Walker, K. Q. Zhu, and P. White, "From dirt to shovels: fully automatic tool generation from ad hoc data," *ACM SIGPLAN Notices*, vol. 43, no. 1, pp. 421–434, 2008.

[10] N. C. Rowe and S. L. Garfinkel, "Finding anomalous and suspicious files from directory metadata on a large corpus," in *International Conference on Digital Forensics and Cyber Crime*. Springer, 2011, pp. 115–130.

[11] D. Scofield, C. Miles, and S. Kuhn, "Fast model learning for the detection of malicious digital documents," in *SSPREW-7*, December 2017.

[12] J. Lundberg, "Classifying dialects using cluster analysis," *Master's thesis, Göteborg University*, 2005.

[13] J. Grieve, D. Speelman, and D. Geeraerts, "A statistical method for the identification and aggregation of regional linguistic variation," *Language Variation and Change*, vol. 23, no. 2, pp. 193–221, 2011.

[14] A. G. Yong, S. Pearce *et al.*, "A beginner's guide to factor analysis: Focusing on exploratory factor analysis," *Tutorials in quantitative methods for psychology*, vol. 9, no. 2, pp. 79–94, 2013.

[15] C. C. Foundation, "Common Crawl," http://commoncrawl.org, 2021, [Online; accessed 11-Mar-2021].

[16] M. Robinson, "Cosheaf representations of relations and dowker complexes," *J Appl. and Comput. Topology*, 2021.

[17] B. C. Arnold and D. Strauss, "Pseudolikelihood estimation: some examples," *Sankhyā: The Indian Journal of Statistics, Series B*, pp. 233–243, 1991.

[18] Plotly, "3d network graphs in python/v3," 2022. [Online]. Available: https://plotly.com/python/v3/3d-network-graph/

[19] igraph, "python-igraph manual," 2020. [Online]. Available: https://igraph.org/python/tutorial/latest/tutorial.html

[20] M. Laphroaig, Ed., *An address to the secret society of POC||GTFO concerning the gospel of the weird machines*, vol. 0x03, March 2014. [Online]. Available: https://unpack.debug.su/pocorgtfo/pocorgtfo03.pdf