Models and Metrics for Mining Meaningful Metadata

Tyler J. Skluzacek[1], Matthew Chen[2], Erica Hsu[3],
Kyle Chard[1,4], and Ian Foster[1,4]

[1] University of Chicago, Chicago IL, USA
[2] University of Illinois at Urbana-Champaign, Champaign IL, USA
[3] Carnegie Mellon University, Pittsburgh PA, USA
[4] Argonne National Lab, Lemont IL, USA
skluzacek@uchicago.edu

Abstract. The increasing volume and variety of science data has led to the creation of metadata extraction systems that automatically derive and synthesize relevant information from files. A critical component of metadata extraction systems is a mechanism for mapping extractors—lightweight tools to mine information from a particular file types—to each file in a repository. However, existing methods do little to address the heterogeneity and scale of science data, thereby leaving valuable data unextracted or wasting significant compute resources applying incorrect extractors to data. We construct an extractor scheduler that leverages file type identification (FTI) methods. We show that by training lightweight multi-label, multi-class statistical models on byte samples from files, we can correctly map 35% more extractors to files than by using libmagic. Further, we introduce a metadata quality toolkit to automatically assess the utility of extracted metadata.

Keywords: Metadata Quality · Extraction · File Type Identification

1 Introduction

The many files accumulated within science and engineering organizations may, both individually and collectively, contain data of great value. However, poor organization and inadequate documentation frequently make these files difficult for users to navigate. In order to promote repository navigability, metadata extraction systems [18, 12, 11, 5, 23] have been developed to automatically populate rich, searchable data catalogs. Metadata extraction systems generally follow a common structure, as illustrated in Figure 1, in which the following steps are performed in order: (A) iterate over all files in a repository; (B) identify the *type(s)* of each file (e.g., free text, tabular, image); (C) invoke one or more *extractors* (sometimes called *parsers*) on each file to obtain metadata; and (D) perform an action with the resulting metadata (e.g., load a search index). However, different metadata extraction systems focus on different use cases, data types, and communities, and therefore apply different approaches for each stage.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_30

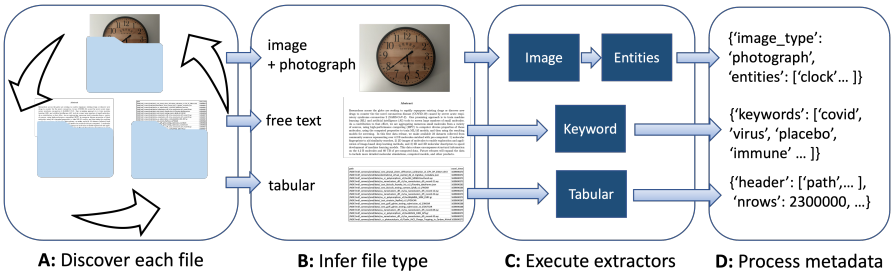


Fig. 1: **Automated metadata extraction steps:** (A) find all files in repository, (B) infer each file's type such that it can be mapped to applicable extractors, (C) execute one or more extractors, (D) post-process metadata.

In our work we focus on extracting metadata from scientific data, an important step for making these complex data navigable and increasing data utility. While much prior research has focused on metadata extraction from personal and enterprise file collections, there is relatively little focus on scientific data, and in particular on the unique challenges posed by these data. For example, the broad nature of scientific inquiry leads scientists to store data in esoteric formats, without regard for schema or file extension; data are often encoded in multi-dimensional file formats that integrate various data types into single files, or spanning several files; and the rise of IoT and decentralized storage has led to data repositories being spread across disparate compute resources.

The growing volume and velocity of scientific data leads us to closely consider the resources used when extracting metadata. Naively applying all extractors to each file is not only inefficient, but may also lead to incorrect or irrelevant metadata. In Figure 2, we illustrate execution times when exhaustively invoking a library of eight extractors on every file in the 428 000-file Carbon Dioxide Information Analysis Center (CDIAC) data set [2]. The figure shows that while most extractors fail quickly, significant compute time is wasted; we estimate that successful invocations consume 130 core hours, whereas applying incorrect extractors (e.g., a NetCDF extractor on a Python script) consumes 670 core hours while returning no valid metadata. When mapping files to extractors, even the most advanced extraction systems do little more than map a mimeType, extension, or byte-regex to a single extractor. However, when scientists create data in bespoke formats or store diverse data types within a single file, these modes of mapping extractors to files often fail.

In this paper, we present an intelligent extractor scheduler for the Xtract metadata extraction system [18] that bridges many of the challenges in applying extractors to science data. While our prior work has focused on issues of scale and decentralization [18], we focus here on directly addressing file diversity by leveraging prior research in file type identification (FTI). We construct statistical learning models that, when used as part of our scheduler, can prioritize the application of extractors to collections of files; thereby maximizing the metadata

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_30

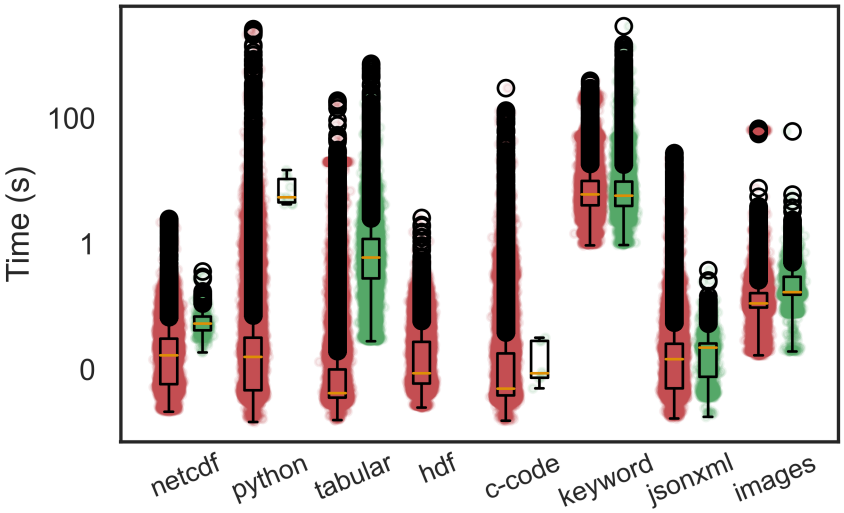


Fig. 2: Box plots (and point distributions) for the invocation of each extractor to each file in CDIAC. The green plots are file-extractor invocations that successfully yield metadata; red are failed invocations (i.e., wasted core hours).

information obtained. Further, we evaluate the efficacy of these methods via a set of automatically derived metadata quality metrics. The contributions of our work are:

- Parameterization and evaluation of FTI methods for metadata extraction.
- Comparative evaluation that shows that our models outperform a state-of-the-art tool (libmagic [1]) in mapping extractors to files by 35%.
- Application of FTI methods on two large, uniquely-diverse scientific data repositories: the heterogeneous Carbon Dioxide Information Analysis Center (CDIAC) and the homogeneous COVID-19 Open Research Dataset (CORD) [24].
- An automated metadata quality analysis toolkit capable of evaluating extracted metadata.

The remainder of this paper is as follows. §2 presents related work in extraction systems and FTI. §3 outlines automated metadata quality metrics. §4 presents our algorithms, learning models, and quality metrics to be evaluated. §5 contains the evaluation of our work on two uniquely diverse scientific data repositories. Finally, §6 summarizes our contributions.

2 Related Work

In this section, we review related work in metadata extraction systems and FTI.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_30

2.1 Metadata Extraction Systems

When evaluating the breadth of open-source metadata extraction systems (as illustrated in Table 1), we observe recurring research gaps: most systems do not cater to the scale and decentralized nature of modern scientific data; none consider the quality of returned metadata; and most have rigid schema constraints (i.e., only process a handful of file types) or manually map file mimeTypes for extractors, and therefore cannot support files of multiple types (e.g., a tabular CSV file with a free text header). To the best of our knowledge, no prior system prioritizes extractors based on the expected value of metadata. While this work strictly focuses on designing an FTI-based extractor scheduler for our system Xtract, prior work illuminates the system design [18, 3] and extractor library [19].

Table 1: Taxonomy of metadata extraction systems. We illustrate differences in systems' mechanisms for scaling extractions (`Parallel`), whether they require the transfer of data from the edge to a centralized compute resource (`Centralized`), their strategy for mapping extractors to files (`Mapping`), whether they provide quality metrics for automatically extracted metadata (`Quality`), and the supported science domains (`Domains`).

System	Parallel	Central	Mapping	Quality	Domain
Tika [12]	Threads	No	extension, mimeType, byte-matches	None	general
Clowder [11]	Cloud	Yes	mimeType	None	general
BDQC [5]	None	Yes	input schema	None	biomedicine
Constellation [23]	Cloud	Yes	input schema	None	general
ScienceSearch [16]	Cluster	Yes	input schema	None	microscopy
Xtract	Cluster, Cloud	No	FTI	Yes	general

2.2 File Type Identification

File type identification (FTI) aims to automatically classify files from inscribed physical contents and is commonly used in digital forensics [15] and malware detection [21]. FTI methods traditionally rely on easily-attainable features from the file (bytes, extension, size). However, science data creates unique challenges as file creators do not adhere to common file extensions, mimeTypes, or schema [20].

FTI methods are crucial to metadata extraction for two reasons: (1) only extractors yielding metadata should be executed on a file, and (2) the multi-output nature of statistical learning algorithms corresponds well with multi-typed files. We consider significant prior work in evaluating and selecting features and models that may work well for the extractor mapping problem [6, 9, 13]. We

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_30

evaluate the performance of our own file type identification methods with metrics from prior FTI work (i.e., train time, precision, recall, F1) and metadata quality metrics, as outlined in the following section.

3 Metadata Quality Determination

Ultimately, the goal of metadata extraction systems is to derive useful metadata; however, current extraction systems do not consider the utility of extracted metadata for either individual files or entire data collections. Metadata quality metrics are thus necessary to illuminate the value of applying a given extractor to a file, and by extension, enables us to evaluate the efficacy of FTI methods and extraction systems. While there is some prior work in metadata quality metrics [8], we specifically seek out metrics to automatically quantify the utility of a metadata corpus. We identify the following metrics that measure various dimensions of utility: yield, completeness, entropy, and readability.

Yield. Metadata yield is the total amount of metadata, measured as the number of bytes of metadata produced. While an imperfect measure, yield is useful for understanding the context of the other metrics, and is easy to obtain. For instance, how do 5 "readable" bytes compare to 1000 that are less readable?

Completeness. Metadata are complete if they contain all possible attributes that could be obtained. In practice, and especially in the presence of diverse schema, some metadata attributes may be left empty. The simplest completeness metric [14] simply divides the number of metadata elements by the total number of elements that could be obtained (i.e., a percentage). We call this metric *simple_completeness* and define it in Equation (1), where N is the number of attributes and $P(i)$ is 0 if the i^{th} metadata attribute is null, and 1 otherwise:

$$simple\_completeness = \sum_{i=1}^{N} \frac{P(i)}{N} * 100 \tag{1}$$

Other researchers [7] have created weighted versions of completeness to account for some attributes exhibiting higher semantic importance than others. They enable multi-tiered importance by including in their completeness score an "absence" penalty for missing metadata elements, where a higher weight corresponds to subjectively more-relevant attributes. Even further, others have accounted for the weighted importance of values in hierarchical attributes [10]. While the aforementioned measures of completeness arguably better represent a human's subjective view of "completeness", it is difficult to have humans manually provide weights, primarily due to the propensity of users to bias higher weights onto items they personally correlate to value [4]. While we plan to properly address (and de-bias) weighting strategies via user study in future work, we use *simple_completeness* as a sufficient and fair proxy-measure in this paper.

Entropy. Metadata entropy [17] is the degree to which metadata presents information that is different from other metadata. A common approach is to apply

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_30

Term Frequency-Inverse Document Frequency (TF-IDF) to determine the entropy of a metadata document. TF-IDF for a metadata document provides an importance score for all words in a document, relative to all metadata documents in the corpus. Scientists [14] have proposed a score built on TF-IDF that produces an entropy score for a metadata document as is shown in Equation (2), where N is the number of text attributes, $attribute_i$ the i^{th} attribute of metadata, and $sum\_tf(attribute_i)$ the sum of TF-IDF scores for a given attribute (in a document):

$$entropy = \log(\sum_{i=1}^{N} sum\_tf(attribute\_i)) \tag{2}$$

Readability Readability measures the ability of humans to semantically interpret metadata. In this work, we leverage the Flesch Index [22]—a document score that compounds the complexity of words and sentences onto a 0–100 scale where documents scoring near 0 are unintelligible to most human readers and those scoring near 100 are broadly understood. For metadata documents in a search index, we ideally give higher semantic weight to metadata containing searchable words, thereby penalizing number-dominated metadata. To accomplish this, we weight the Flesch index by the proportion of characters (n_char) that are not numbers (n_num): $W_s = (1 - \frac{n\_num}{n\_char})$. To account for decimal points potentially misrepresenting the ends of sentences in numeric metadata, we remove all mid-numeric decimal points prior to tokenizing. We then define our weighted Flesch index $WFlesch$, where n_word, n_sent, n_syl are the number of words, sentences, and syllables, as follows:

$$WFlesch = \overbrace{(206.835 - 1.015(\frac{n\_word}{n\_sent}) - 84.6(\frac{n\_syl}{n\_word}))}^{\text{original Flesch Index}} * W_s \tag{3}$$

4 Methodology

We now describe our process for using statistical learning models to identify applicable extractors for each file in a science repository. Specifically, we describe how we label data, generate features, select models, and leverage model outputs as input to the extraction scheduler.

Label Generation. We first create a library of ground-truth labels for all files in both science repositories. To this end, we exhaustively apply each extractor to every file, and record (1) the metadata returned by the extractor, and (2) the time taken to execute the extractor. The file is assigned a label for each extractor that, when applied to it, returns nonempty metadata. If a file receives no labels, we use heuristic methods to determine whether a file might be compressed, a binary executable, or empty—and if none of those—it receives a label of "unknown." For supervised model training, we place each possible type label for a given file $f \in F$ into a label vector $L(f) = [is(f, t_0), is(f, t_1)...is(f, t_m)]$ where $is(f, t_i)$ is 1 if f is of type $t \in T$, and 0 if not.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_30

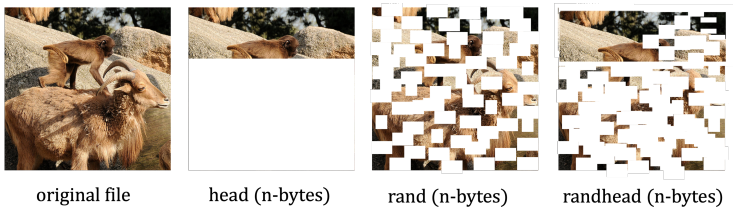


Fig. 3: Feature illustration for head, rand, and randhead.

Feature selection. We create input feature vectors containing (i) file size and (ii) 16–512 byte samples from the file. As illustrated in Figure 3, we fetch byte samples from the following locations in the file: the header (head), randomly throughout (rand), or a combination of both (randhead).

Model selection. We train models to accomplish the following: given a file $f \in F$, we want to train a model $m \in M$ such that $m(f)$ generates a probability distribution $P(f) = [p(f, e_1), p(f, e_2), ..., p(f, e_n)]$, where $p(f, e)$ is the probability that f should map to extractor $e \in E$. We explore multiple statistical learning models: logistic regression (logit), random forests (rf), and support vector classification (svc). We evaluate model performance primarily in the form of F1 score (which measures overall model performance) and recall (the percentage of correct file-extractor mappings identified) on weighted multi-class probability distributions generated by each model. We also consider model training time. To account for potential overfitting, we evaluate models on both imbalanced (all data) and balanced (subset of the data) classes.

Extraction Scheduler. Our primary goal is to design an extraction system that converts FTI model outputs into a schedule of file/extractor pairs to execute. We first train lightweight regressions that use a file's size to predict metadata yield $Y(e, size(f))$ and extraction time $T(e, size(f))$. We select our regression based on which has the better correlation score between a linear and nonlinear model [25], and fit the corresponding model. Given our probability vector of file-extractor mappings, $P(f) = [p(f, e_1), p(f, e_2), ..., p(f, e_n)]$, the size of a file $size(f)$, and a $+1$ Laplace smoothing constant, we introduce an objective function to compute predicted metadata yield over time $\alpha(f, e)$:

$$\alpha(f, e) = log(\frac{Y(e, size(f)) * p(f, e) + 1}{T(e, size(f)) + 1}) \tag{4}$$

We prioritize extractor execution by loading a priority queue in descending order of alpha score (i.e., the system maximizes expected metadata yield over time).

5 Evaluation

We analyze the feature and model performance of the FTI methods and compare with libmagic [1]. We then examine metadata quality when using our scheduler.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_30

Science repositories. We evaluate our approach in the context of two distinct scientific repositories. We primarily focus on the Carbon Dioxide Information Analysis Center (CDIAC) a climate science dataset that represents a multi-group conglomeration of carbon dioxide data. We copied these data from their now-defunct FTP server in 2017. These data, whose extensions we visualize as a treemap in Figure 4, have a high degree of variety—there are over 150 unique file extensions spanning 428 000 files, and many of the files are in difficult-to-parse formats (e.g., deprecated Windows installers, Hadoop error logs, and desktop shortcuts) [20]. We include both the unedited file formats consisting of many compressed files (e.g., *.Z*), and also the decompressed contents. We also examine the more-homogeneous COVID-19 Open Research Dataset (CORD) containing 517 000 JSON-formatted COVID-19 research papers spanning 2019–2021.

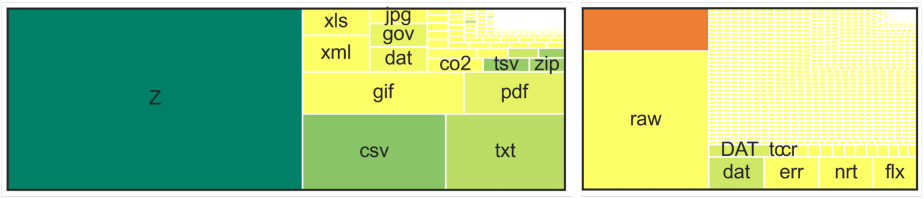


Fig. 4: **Treemaps of CDIAC:** (left) the unedited repository, (right) all decompressed files. Each box's *area* is the proportion of files of that extension, and *darkness* is the relative total size (darker=bigger). The orange box on the right represents files with no extension.

Experimental Testbed. We perform our experiments on ALCF Theta, an 11.7-petaflop Cray XC40 supercomputer with second-generation Intel Xeon Phi "Knight's Landing" (KNL) processors. Each node has a 64-core processor and 166 GB MCDRAM, 192 GB DDR4 RAM, with a shared a Lustre file system.

5.1 FTI Modeling

Features. We first want to find the best byte structure (head, rand, randhead) and number of bytes (16–512) to use as features in our analysis. For all experiments, we use a standard 70%/30% train/test split. Figure 5 shows the range of model scores for the different byte structures on each of the three model types: logit, rf, and svc. We see that, for the CDIAC data, the head bytes outperform rand and randhead in every statistical metric. To investigate whether there is significant benefit beyond 512 head bytes, we compare F1 improvements when doubling from 16 to 32, and 256 to 512, respectively. The relative F1 difference when increasing from 16 to 32 bytes for (logit, rf, svc) is (+1.0, +0.6, +0.7), but the difference between 256 and 512 bytes is only (+0.2,+0.1,-1.9). Therefore, we use 512 head bytes, since additional bytes would likely have marginal benefit. As shown in Table 2, CORD can be processed well by any of the feature configurations.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_30

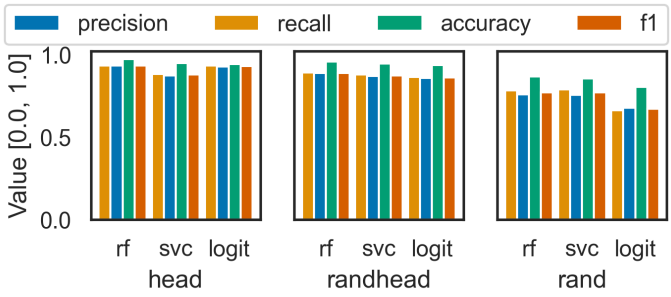


Fig. 5: Model scores for multiple 512-byte feature configurations (CDIAC).

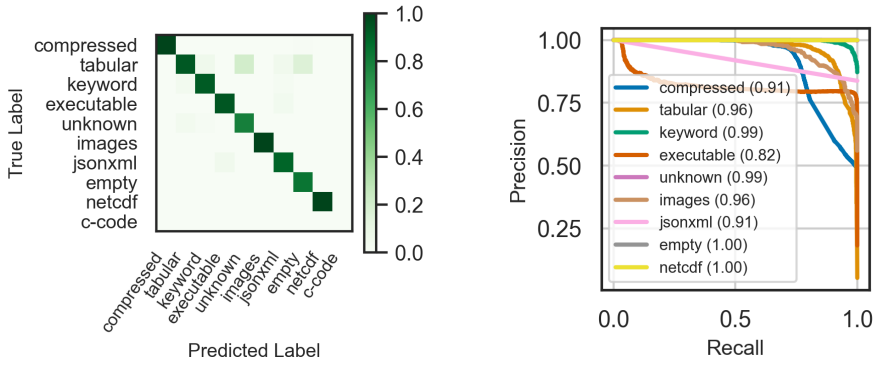
Table 2: Model performance for 16 and 512 bytes for logistic regression (logit), random forests (rf), and support vector classifier (svc) on CDIAC and CORD

Repository	Header Bytes	Model	Train Time (s)	Precision	Recall	F1 Score
CDIAC	16	logit	403	0.839	0.836	0.837
		rf	2.29	0.890	0.896	0.893
		svc	1010	0.856	0.867	0.861
	512	logit	1140	0.930	0.936	0.933
		rf	4.50	0.939	0.938	0.938
		svc	9240	0.875	0.885	0.880
CORD	16	logit	17.0	1.00	1.00	1.00
		rf	3.56	1.00	1.00	1.00
		svc	418	1.00	1.00	1.00
	512	logit	183	1.00	1.00	1.00
		rf	4.25	1.00	1.00	1.00
		svc	464	1.00	1.00	1.00

Models. To avoid overfitting, we train our models on both imbalanced and balanced classes in CDIAC. The imbalanced class (all-of-CDIAC) experiments shown in Figure 6 shows that the 512B random forests model can adequately identify most file types in the CDIAC repository. The confusion matrix shows that the model can effectively identify the top hierarchical type for a file, and the multiclass-weighted PR curve shows that, overall, we see high recall, regardless of precision, for each label type. Interestingly, one can see in both diagrams that the most-difficult class for the model to identify is the "unknown" class.

As the imbalanced model is likely overfit to the larger classes, we propose an experiment on balanced classes; we create a balanced subset of CDIAC by randomly selecting 200 files from each class (and omitting those classes with fewer than 200 files). In Figure 7, the confusion matrix shows that our model can efficiently select the top type of a file, whereas the multiclass-weighted PR curve shows that we see fairly high precision at all levels of recall. Interestingly in the balanced case, we see that the model has added difficulty in selecting

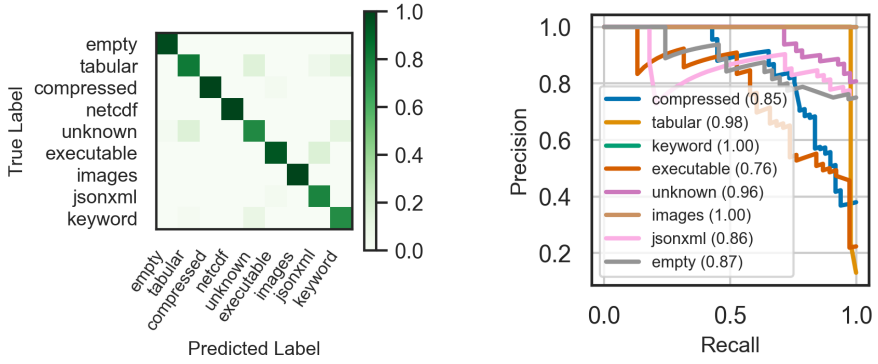
ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_30



(a) Confusion Matrix (n=214 315)

(b) Precision-Recall Curve (Area Under Curve)

Fig. 6: **Imbalanced Classes (CDIAC):** confusion matrix (prediction-normalized) and precision-recall curve (multi-class weighted) for random forests model trained on 512 head bytes.



(a) Confusion Matrix (n=1800)

(b) Precision-Recall Curve

Fig. 7: **Balanced Classes (CDIAC):** confusion matrix (prediction-normalized) and precision-recall curve (multi-class weighted) for random forests model trained on 512 head bytes.

.

tabular files, likely explained by the high overlap of files correctly mapping to multiple extractors.

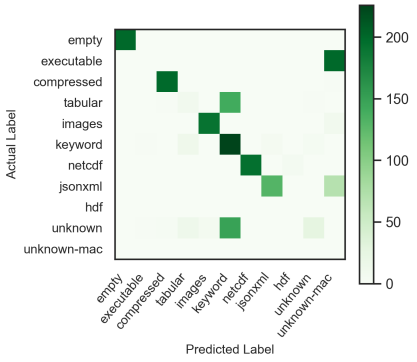
Finally, to study how the model performs when individual files contain multiple content types, we analyze the output probability distributions for all multi-typed files, and investigate whether each type is represented at the top of the probability distribution. In Table 3, we observe that, for CDIAC, most multi-

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_30

typed files share a type with the keyword extractor, and the model identifies the keyword type 80% of the time and the other type 96% of the time.

Table 3: Analysis of files of both types Type 1 and Type 2, and how many of each type are included in the top-2 entries of the probability distribution.

Repository	Type 1	Type 2	Count	Type 1 Included	Type 2 Included
CDIAC	keyword	tabular	12 878	10 966	12 415
		jsonxml	3282	1954	3109
		netcdf	252	205	252
		c-code	8	8	3
		python	3	3	3
	tabular	python	7	7	7
CORD	jsonxml	keyword	517 900	517 900	517 900



(a) Confusion Matrix

Type	Pr.	Re.	F1
empty	1.00	1.00	1.00
executable	0.00	0.00	0.00
compressed	0.98	1.00	0.99
tabular	0.25	0.06	0.10
images	0.96	0.96	0.96
keyword	0.44	0.91	0.59
netcdf	1.00	0.97	0.98
jsonxml	0.96	0.65	0.78
unknown	0.84	0.13	0.23
unkn.-mac	0.00	0.00	0.00

(b) Precision, Recall, F1

Fig. 8: **Libmagic (CDIAC):** confusion matrix and performance metrics for mapping extractors to files using the libmagic FTI tool.

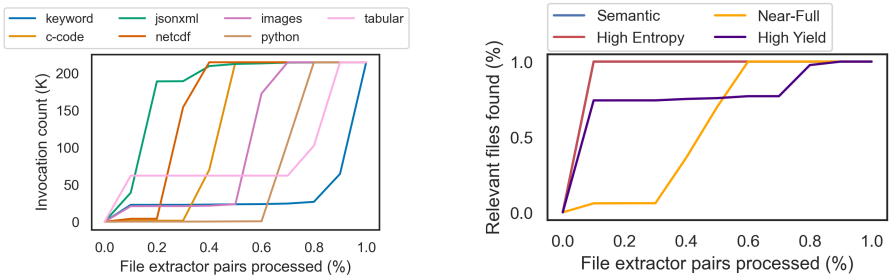
Libmagic Comparison. We next compare our approach to the libmagic FTI tool. As libmagic types do not directly map to our extractor library, we manually map libmagic outputs to our types. Some mappings are obvious (e.g., empty:empty, compress'd:compressed) while others required consulting libmagic documentation (e.g., data:unknown). We compare each libmagic output to our extractor labels, and show the result in Figure 8. Overall, libmagic performs significantly worse than our FTI methods, as it consistently misclassifies tabular

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_30

and keyword data. Even in this favorable experiment, libmagic only accurately identifies 65% of files (cf. our FTI methods correctly identify 88%).

5.2 Extractor Scheduler and Metadata Analysis

We next evaluate the extraction patterns and metadata output over time when the scheduler uses the predicted probability vectors for each file. Figure 9a shows the extractor executions over all possible file-extractor pairs. Given that the scheduler prioritizes metadata yield over time, we notice that extractors that either succeed quickly (jsonxml, images) or those that produce large quantities of metadata (tabular) have initial spikes within the first 10% of invocations.



(a) Extractor invocations over time

(b) Relevant files found over time

Fig. 9: **Scheduler Analysis:** (a) extractor invocations over percentage of file-extractor pairs processed; (b) percentage of total quality files discovered over all file-extractor pairs processed.

To measure the relative utility of metadata extraction, we count "useful" metadata documents as defined by the quality metrics: `semantic` metadata are those that contain searchable words (measured as files with nonzero readability scores), `near-full` metadata contain complete data (measured as files with over 50% completeness), `high entropy` metadata add unique information to the corpus (measured as files with nonzero entropy), and `high yield` metadata exceed 500 bytes. Figure 9b shows that maximizing yield over time naturally prioritizes extracting semantically searchable and high entropy metadata. Intuitively this makes sense as semantic metadata are often larger than average (containing many words), and therefore provide high yield. Therefore, we see that this scheduler could add significant value for organizations looking to create a semantically searchable index with high information content on a limited compute budget.

For purposes of illustrating potential quality bias across extractors, we illustrate in Figure 10 the observed metrics of all CDIAC metadata. We observe that different extractors generate unique quality profiles: tabular metadata exhibit high readability and entropy, keyword metadata exhibit high readability,

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_30

and hdf and image metadata exhibit high completeness. In future work, we will study how extractor quality profiles can be used in an extraction scheduler.

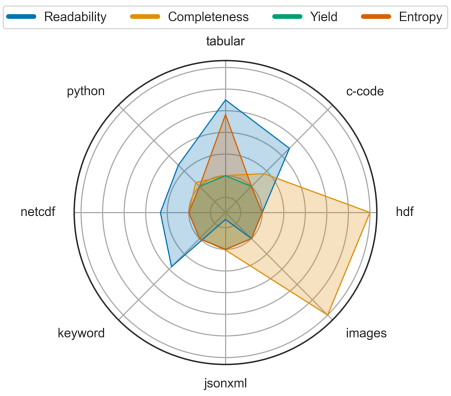


Fig. 10: Spider plot representation of successful metadata extraction metrics on CDIAC. The distance between the center and each extractor is the log-scale range of each metric, and the placement of the colored line represents the median of the corresponding metric.

6 Conclusion

Accurate and performant metadata extraction is dependent on accurate methods for mapping extractors to files; however, traditional methods are not conducive to the wide, heterogeneous variety of science file formats. We introduce several file type identification methods that use lightweight byte-features from files and various machine learning models to predict scientific file types. We use these models to create a scheduler for the Xtract metadata extraction system, enabling Xtract to prioritize application of extractors to files. Further, we introduce several metrics designed to quantify the utility of metadata, and by extension, the usefulness of the extractor scheduler.

Acknowledgements

We gratefully acknowledge Takuya Kurihana (University of Chicago) for sharing his machine learning expertise. This work is supported in part by the National Science Foundation under Grants No. 2004894 and 1757970, and used resources of the Argonne Leadership Computing Facility.

References

1. Libmagic(3) - linux man page (Nov 2009), https://linux.die.net/man/3/libmagic
2. Cdiac (Mar 2018), https://cdiac.ess-dive.lbl.gov/
3. Chard, R., Babuji, Y., Li, Z., Skluzacek, T., Woodard, A., Blaiszik, B., Foster, I., Chard, K.: Funcx: A federated function serving fabric for science. In: Proceedings of the 29th International symposium on high-performance parallel and distributed computing. pp. 65–76 (2020)
4. Deng, H., Runger, G., Tuv, E.: Bias of importance measures for multi-valued attributes and solutions. In: Int'l Conf. on Artificial Neural Networks. pp. 293–300 (2011)

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_30

5. Deutsch, E.W., Kramer, R., Ames, J., Bauman, A., Campbell, D.S., Chard, K., Clark, K., D'Arcy, M., Dinov, I.D., Donovan, R., et al.: BDQC: A general-purpose analytics tool for domain-blind validation of big data. bioRxiv p. 258822 (2018)
6. Gopal, S., Yang, Y., Salomatin, K., et al.: Statistical learning for file-type identification. In: Int'l Conf. on Machine Learning and Applications. pp. 68–73 (2011)
7. Hughes, B.: Metadata quality evaluation: Experience from the open language archives community. In: Int'l Conference on Asian Digital Libraries. pp. 320–329. Springer (2004)
8. Király, P.: Measuring Metadata Quality. Ph.D. thesis, Georg-August-Universität Göttingen (06 2019). https://doi.org/10.13140/RG.2.2.33177.77920
9. Li, W.J., Wang, K., Stolfo, S.J., Herzog, B.: Fileprints: Identifying file types by n-gram analysis. In: IEEE SMC Information Assurance Workshop. pp. 64–71 (2005)
10. Margaritopoulos, M., Margaritopoulos, T., Mavridis, I., Manitsaris, A.: Quantifying and measuring metadata completeness. Journal of the American Society for Information Science and Technology **63**(4), 724–737 (2012)
11. Marini, L., Gutierrez-Polo, I., et al.: Clowder: Open source data management for long tail data. In: Practice and Experience on Adv. Research Computing (2018)
12. Mattmann, C., Zitting, J.: Tika in Action. Manning Publications Co., USA (2011)
13. McDaniel, M., Heydari, M.H.: Content based file type detection algorithms. In: 36th Annual Hawaii Int'l Conference on System Sciences. pp. 10–pp. IEEE (2003)
14. Ochoa, X., Duval, E.: Automatic evaluation of metadata quality in digital repositories. Int'l Journal on Digital Libraries **10**(2-3), 67–91 (2009)
15. Poisel, R., Tjoa, S.: A comprehensive literature review of file carving. In: Int'l Conference on Availability, Reliability and Security. pp. 475–484. IEEE (2013)
16. Rodrigo, G., Henderson, M., et al.: ScienceSearch: Enabling search through automatic metadata generation. In: 14th Int'l Conf. on e-Science. pp. 93–104 (2018)
17. Shannon, C.E.: A mathematical theory of communication. ACM SIGMOBILE Mobile Computing and Communications Review **5**(1), 3–55 (2001)
18. Skluzacek, T., Wong, R., Li, Z., Chard, R., Chard, K., Foster, I., Chard, K.: A serverless framework for distributed bulk metadata extraction. In: 30th Int'l Symp on High-Performance Parallel and Distributed Computing (2021)
19. Skluzacek, T.J., Chard, R., Wong, R., Li, Z., Babuji, Y.N., Ward, L., Blaiszik, B., Chard, K., Foster, I.: Serverless workflows for indexing large scientific data. In: Int'l Workshop on Serverless Computing. pp. 43–48 (2019)
20. Skluzacek, T.J., Kumar, R., Chard, R., Harrison, G., Beckman, P., Chard, K., Foster, I.T.: Skluma: An extensible metadata extraction pipeline for disorganized data. In: IEEE 14th Int'l Conference on e-Science. pp. 256–266. IEEE (2018)
21. Tabish, S.M., Shafiq, M.Z., Farooq, M.: Malware detection using statistical analysis of byte-level file content. In: ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics. pp. 23–31 (2009)
22. Talburt, J.: The Flesch index: An easily programmable readability analysis algorithm. In: Int'l Conference on Systems Documentation. pp. 114–122 (1986)
23. Vazhkudai, S.S., Harney, J., et al.: Constellation: A science graph network for scalable data and knowledge discovery in extreme-scale scientific collaborations. In: IEEE Int'l Conference on Big Data. pp. 3052–3061 (2016)
24. Wang, L.L., Lo, K., et al.: Cord-19: The covid-19 open research dataset. arXiv:2004.10706 (2020). https://doi.org/10.48550/ARXIV.2004.10706
25. Wang, Y., Li, Y., et al.: Efficient test for nonlinear dependence of two continuous variables. BMC Bioinformatics **16**(1), 1–8 (2015)

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_30