From why-provenance to why+provenance: Towards addressing deep data explanations in Data-Centric AI

Paolo Missier[1,*], Riccardo Torlone[2]

[1]University of Birmingham, School of Computer Science, Birmingham, UK
[2]Università Roma Tre, Roma, Dipartimento di Ingegneria, Italy

Abstract
In this position paper we discuss the problem of exploiting data provenance to provide explanations in data-centric AI processes, where the emphasis of model development is placed on the quality of data. In particular, we show how a classification of the main operators used in the data preparation phase provides an effective and powerful means for the production of increasingly detailed explanations at the needed level of data granularity.

Keywords
Data-centric AI, Data Engineering pipeline, Data Provenance

1. Introduction

In provenance theory, the notion of *why-provenance* has been introduced primarily in the context of relational models and algebra with reference to the set of tuples in source relations that contribute to producing the results of a (SQL) query. These have been known as witness tuples [1, 2] and define the *lineage* of a tuple that appears in the result of a query. The clear semantics associated with relational algebra operators made it possible to develop formal, elegant models for representing *why*-provenance, and its extensions to *how*-provenance [3]. Further extensions have subsequently been developed, for instance to capture the provenance of results of aggregated queries [4].

A parallel strand of research focused on the definition of provenance as it applies to datasets that undergo a series of transformations, where arbitrary operators are typically arranged into a Directed Acyclic Graph topology. Provenance in this setting can itself be expressed as a graph of data derivations, where each derivation is mediated by one operator. This became known as "coarse-grained" provenance [5], in contrast to the tuple-level provenance grounded in the relational framework. As the term suggests, the "black-box" nature of the operators makes granular provenance hard to determine directly [6]. This is because why- and how-provenance are grounded in the precise semantics of the query operators, which is not available when using arbitrary data transformers. The result is a provenance graph that is limited to dataset-level derivations, i.e., through each of the processors. Existing approaches that attempt to circumvent this limitation and reconstruct "high-fidelity" provenance, using system-level events recorded

SEBD 2024: 32nd Symposium on Advanced Database Systems, June 23-26, 2024, Villasimius, Sardinia, Italy
*Corresponding author.
p.missier@bham.ac.uk (P. Missier); riccardo.torlone@uniroma3.it (R. Torlone)
0000-0002-0978-2446 (P. Missier); 0000-0003-1484-3693 (R. Torlone)

during execution ("provenance record and replay" [7]), are designed to apply to completely unstructured processes and may incur significant computational overhead.

In this position paper, we focus on the provenance of data transformations that occur in the context of Data Science, where dataflow-structured data analytics pipelines are common, and where their elements are not simple relational operators (as was assumed e.g. in [8]), but they are not completely arbitrary, either. In this middle-ground setting, we consider the problem of using provenance to generate explanations that justify and account for the observed transformations. Looking at the semantics of the processors involved in the pipeline reveals a range of automatic provenance-generation capabilities, with relational, how-provenance on one extreme, and arbitrary data manipulation code, on the other.

We suggest that an interesting region within this spectrum is occupied by a new generation of operators, which are defined in the context of so-called Data-Centric AI (DCAI). These are sophisticated operators specifically designed to produce training sets from raw datasets, and where data processing is often interleaved with model training, in an iterative fashion. While their semantics is not formally defined, these tend to fall into a few categories, for example, data transformations such as incremental data cleaning, data augmentation, for instance through upsampling algorithms, and data selection, including feature selection and elimination of redundant data points.

In the rest of this paper, we present exemplar use cases of DCAI processes taken from recent literature, and propose a categorisation of operators that extends from our previous work [9], showing how the use cases relate to these categories. We then discuss options for generating "provenance narratives" to describe these operators' behavior. Somewhat provocatively, we refer to these narratives as *why+provenance*, to indicate that they can be used to answer "why" type questions with reference to complex but well-defined data transformations.

2. Example use cases

We present four use cases, where data transformation and data selection exhibit two kinds of complexities, either they are entangled with the modelling itself, or they implement some bespoke data manipulation strategy that is not captured by typical data processing operators.

2.1. Model-driven incremental data cleaning

ActiveClean [10] is one of several incremental data cleaning algorithms, surveyed in [11], targeted specifically at training sets. It provides a good example of an iterative approach designed to progressively clean a "dirty" training set D by balancing the cost of selecting and cleaning items in D, with the benefits of learning a usable model, despite being trained on a training set that is still partially dirty [12]. A "dirty" multidimensional data point is one where one or more of its components is inaccurate, for instance a wrong numerical figure, or a wrongly spelled name. As observed in [11], these dirty data lead to a sub-optimal training process, where the model parameters are optimised, but for a loss function that corresponds to a misleading training set, potentially rendering the model useless in practice.

Cleaning is generally an expensive operation, especially when it requires manual inspection. The idea behind ActiveClean is to start by selecting a subset of dirty data points from D, manually clean them to generate D_1, and use D_1 to retrain the model. This is repeated until a stop condition is reached, producing a sequence $D \rightarrow D_1, D_2, \ldots, D'$ of training sets. There is an assumption that the dirty/clean status of a data point can be automatically detected, and that one can optimise the procedure by choosing the data points to be cleaned that most affect the model, with the goal to minimise the number of iterations.

2.2. Training set debugging

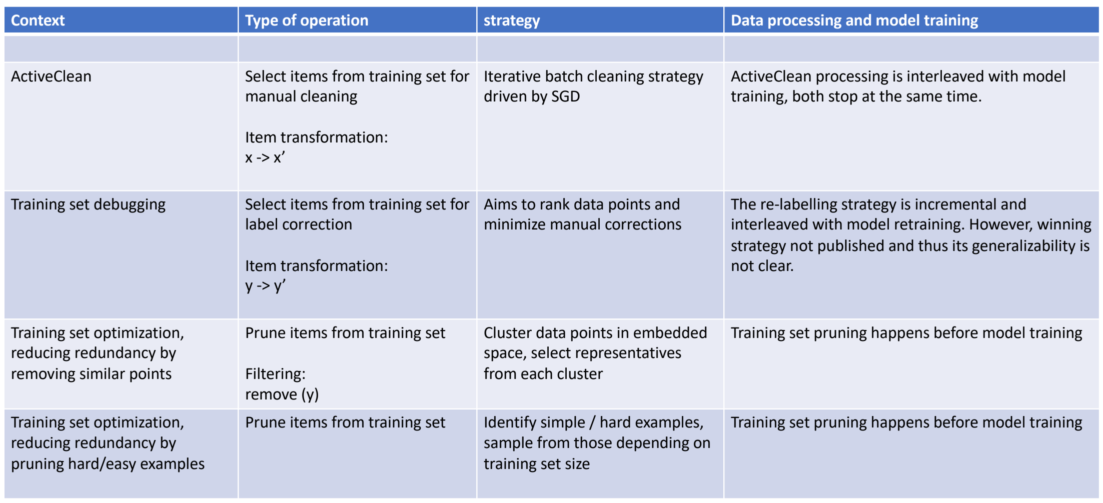
This use case is similar to the previous one, but here we assume that the ground truth labels associated with some of the data points, as opposed to the features, may be incorrect. The challenge, as it was presented by the DataPerf group [13] (https://www.dataperf.org/) as part of ML Commons (https://github.com/mlcommons), is to devise a strategy by which a sufficiently accurate model can be trained by correcting the fewest possible labels. Specifically, the challenge used annotated image data from the OpenImage V7 dataset (https://storage.googleapis.com/openimages/web), which contains millions of images with various levels of annotations (bounding boxes, relationships, image-level, point-level labels, etc.). Given a training set D^{tr} taken from OpenImage with perfect annotations and a predefined classification task T, a model $M(D^{tr})$ is trained and its performance P (according to some agreed upon metric) is used as a benchmark for the challenge. Some of the labels in D^{tr} are then randomly corrupted, leading to a noisy set D_n. The performance P_n of a model obtained by training a classifier for T using D_n will in general be sub-optimal, $P_n < P$. The challenge is to devise a strategy for selecting the smallest possible subset $D'_n \subset D_n$ such that, by correcting the labels in D'_n and then retraining, the new performance will approximate P within some predefined threshold τ: $P - P_n < \tau$.

2.3. Training set optimisation

The next two use cases are motivated by the well-known "power laws" observation, common in deep learning applications, that model performance (test loss) correlates positively with training set size according to a power law [14]. However, increasing training set sizes ultimately incurs diminishing returns in terms of loss reduction. This motivates trying to prune D to reduce its size. Two approaches stand out, both proposed by the same researchers.

Firstly, in [15] the idea is to map D to an embedded space, using pre-trained foundation models, then cluster all data points in that space using a standard clustering algorithm (k-means). Neighbouring points within each cluster (according to some distance metrics) are considered redundant and candidates for pruning, resulting in the final D'.

The second approach [16] is based on the concept of data points that are *easy* or *hard* to learn from. Two main results underpin the pruning method. Firstly, the authors claim that the difficulty of each data point is proportional to the distance of that point from the centroid of a k-means cluster, where the clustering is performed in an embedded space. Once the points are ranked in terms of their difficulty, the second claim is that hard examples should be preserved for large training sets, and conversely, the easier ones should be preferred for smaller training sets (details are in the paper). It should be noted that the experiments supporting this claim were only performed using a dedicated self-supervised model pre-trained on ImageNet, and may not generalise well to other contexts.



Context	Type of operation	strategy	Data processing and model training
ActiveClean	Select items from training set for manual cleaning Item transformation: x -> x'	Iterative batch cleaning strategy driven by SGD	ActiveClean processing is interleaved with model training, both stop at the same time.
Training set debugging	Select items from training set for label correction Item transformation: y -> y'	Aims to rank data points and minimize manual corrections	The re-labelling strategy is incremental and interleaved with model retraining. However, winning strategy not published and thus its generalizability is not clear.
Training set optimization, reducing redundancy by removing similar points	Prune items from training set Filtering: remove (y)	Cluster data points in embedded space, select representatives from each cluster	Training set pruning happens before model training
Training set optimization, reducing redundancy by pruning hard/easy examples	Prune items from training set	Identify simple / hard examples, sample from those depending on training set size	Training set pruning happens before model training

Figure 1: Summary of data interventions for the example use cases

3. Representing provenance at multiple levels of detail

With reference to the examples just presented, we would like to provide provenance support for answering the following types of questions. Firstly, which data transformation were applied to raw input dataset(s) to generate the final training set used for modelling? Secondly, which of the individual data items where affected by each of the transformations, and what was the effect? And thirdly, *why* was a specific data item chosen for transformation or inclusion/exclusion, and in the case of transformations, how was a specific new value chosen? These questions address issues of reproducibility, specifically when the operator is part of a processing pipeline, and explainability, both at the level of entire training set and of individual data points.

Viewed at a high level, the examples fall into the broad category of data transformation: $D \rightarrow D'$, and selection: $D' \subset D$. At this level, it is straightforward to record the provenance of D' as a derivation from D, which is mediated by some abstract activity A that represents the cleaning or pruning operations. Using the formal notation provided by the PROV data model [17], this can be written simply as:

$$\begin{aligned}
&\texttt{activity}(A) && \text{\# an activity represents a data operator}\\
&\texttt{entity}(D), \texttt{entity}(D') && \text{\# entities represent datasets}\\
&\texttt{Used}(A, D) && \text{\# } A \text{ consumes input dataset } D\\
&\texttt{WasGeneratedBy}(D', A) && \text{\# } A \text{ produces output dataset } D'\\
&\texttt{WasDerivedFrom}(D', D) && \text{\# data-data derivations}
\end{aligned}$$

Fig. 2(a) shows a corresponding graph representation for these derivations. This high-level provenance is not very informative, however, if we want to account for how A operates on each data item. In the first two examples, A performs 1-1, item-wise transformations, i.e., $x \in D \rightarrow x' \in D'$ where either $x' = x$ or x' is a clean version of x. Our PROV notation can be

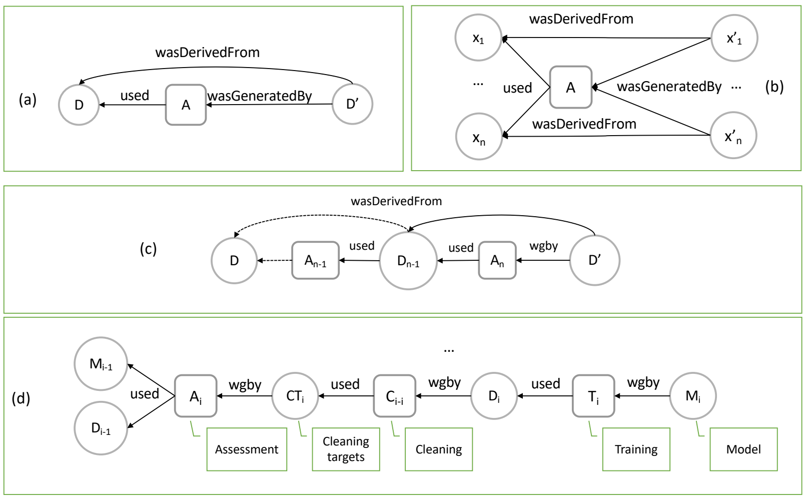


Figure 2: Provenance patterns for the examples in the text

extended to account for this more granular level, as follows (see also Fig. 2(b)):

$\{\texttt{entity}(x)\}_{x \in D}$ # items in D

$\{\texttt{entity}(x')\}_{x' \in D'}$ # items in D'

$\texttt{activity}(A)$ # A is still an atomic perator

$\{\texttt{Used}(A, x)\}_{x \in D}$ # x that have been affected by A

$\{\texttt{WasGeneratedBy}(x', A)\}_{x' \in D'}$ # and their new values x'

$\{\texttt{WasDerivedFrom}(x', x)\}$ # data-data derivations

Using these assertions, one can reconstruct the derivations for any data item, from the initial D to the final D', along a whole sequence of operators, through simple traversal queries.

In this simple example, the notation is used to represent item-wise transformations, i.e., by creating instances for each `Used`, `wasGeneratedBy`, and `wasDerivedBy` relationship for corresponding items x, x'. Note however, that this can also be used, more generally, to capture M-N transformations, for example to represent the effects of data imputation based on aggregate statistics that affect multiple data points simultaneously. This can be achieved by adding relationship instances as needed. For example, when a single value $y \in D$ is used to produce multiple values $x'_1, \dots, x'_n$, the derivation can be written as $\{\texttt{WasDerivedFrom}(x'_i, y)\}_{i:1,n}$.

We can further account for the incremental nature of cleaning in ActiveClean, by breaking down A into $A_i \dots A_n$ and explicitly representing n iterations (Fig. 2(c)):

$\{\texttt{entity}(D_i)\}_{i:0}^{n}$ # each D_i is the result of one iteration

$\{\texttt{activity}(A_i)\}_{i:1}^{n}$ # A_i represents one cleaning round

$\{\texttt{WasDerivedFrom}(D_i, D_{i-1})\}_{i:1}^{n}$ # data-data derivations for one iteration

$\{\texttt{Used}(A_i), D_{i-1},\}_{i:1}^{n}$ # A_i consumes D_{i-1}

$\{\texttt{WasGeneratedBy}(D_i, A_i)\}_{i:1}^{n}$ # A_i produces D_i

A similar formal notation can be used to represent the selection operations in the second two examples, both at a dataset level and at item level (details omitted for brevity).

In previous work [9] we have shown how these representations can be automatically generated in the common case where the operators are implemented in Python / Pandas / scikit-learn, and D are Pandas dataframes. We also presented a prototype-level tool [18] to show that item-level provenance within each pair D, D' can be accurately inferred by observing the differences in schema and content between D and D'.

These results effectively address the first two of the three questions above. Addressing the third "why" question is harder, however, as it requires capturing the internal processing logic of complex operators, at some level of abstraction. For instance, the *why+provenance* of a data item that was cleaned using ActiveClean would include not only the before/after values, but also an explanation of why that item was selected for cleaning. Similarly, the *why+provenance* of an item that was included/discarded as part of a training set optimisation process would provide an insight into why that particular item was identified, for instance as being an easy/hard example, or as being redundant.

Our position is that this new level of detail will become increasingly relevant, as Data Science pipelines expand their scope from well-understood operators, to sophisticated black-box algorithms that affect training sets in complex ways.

This is the focus of the rest of the paper. In the next Section we summarise and extend the high-level classification of operators from [9], as a starting point for discussing options to describe more DCAI-specific patterns.

4. A classification of pipeline operators

Based on the analysis of the main Python libraries used in data science, we have observed that the majority of pre-processing operations, including the most used in practice, can be implemented by combining a rather small set of basic operators of data manipulation over datasets belonging to four main classes, as follows.

Data reductions: operations that take as input a dataset D and reduce its size by eliminating rows or columns from D. These are simple extensions of two well-known relational operators: the *(conditional) projection* of D on a set of features in S, given a boolean condition C, is the dataset obtained from D by including only the columns of D that satisfy C; and the *selection* of D, given a boolean condition C, is the dataset obtained from D by including the rows of D satisfying C.

Data augmentations: operations that take as input a dataset D on a schema S and increase the size of D by adding rows or columns to D. These two operators allow the addition of columns and rows to a dataset, respectively: the *vertical augmentation* of D to Y using a function f over a set X of features of D, is obtained by adding to each row of D a new set of features whose values are obtained by applying f to the features in X; and the *horizontal augmentation* of D using an aggregative function f is obtained by adding one or more new rows to D obtained by first grouping over a set of features of D and then by applying f to each group.

Data transformation: the *transformation* of a set of features X of D using a function f is obtained by applying f to all the values occurring in X.

Data fusion: operations that take as input two datasets D_1 and D_2 and combine them into a new dataset D: the *join* of the two datasets based on a boolean condition C is the dataset obtained by applying a standard join operation (inner, (left/right/full) outer) based on the condition C; the *append* of the two datasets is the dataset obtained by appending D^2 to D^1 and possibly extending the result with nulls on the mismatching columns.

Figure 1 reports some common data pre-processing operators and the way in which they can be implemented by combining the above basic operators.

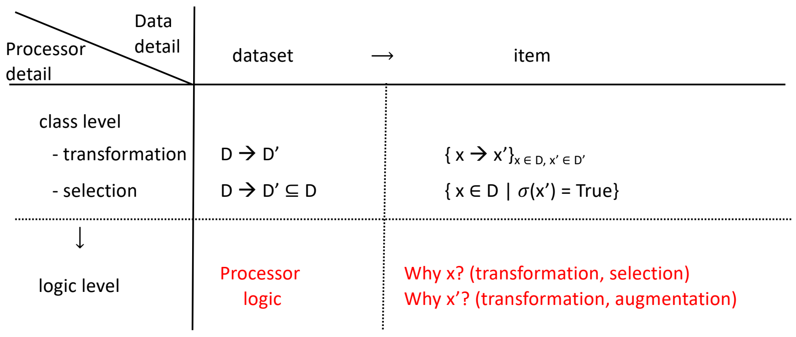
Pre-processing Operations	Basic Operators	Description
Feature Selection	Conditional projection	One or more features are removed.
Instance Drop	Selection	One or more records are removed.
Feature Augmentation	Vertical Augmentation	One or more features are added.
Space Transformation	Vertical Augmentation + Conditional projection	New features are derived from old features, which can be later dropped.
One-hot encoding	Vertical Augmentation + Conditional projection	New features are derived from old features, which can be later dropped.
Instance Generation	Horizontal Augmentation	One or more records are added.
Imputation, Data Type Conversion, Renaming, Normalization, Scaler, Encoding	Transformation	Values are modified using various functions.
Dimensionality Reduction	Transformation + Conditional projection	Some features are modified, others are removed.
Integration, Cartesian Product	Join	Two or more datasets are combined based on a common attribute or key.
Concatenate	Append	Two datasets are combined by taking their union.

Table 1
common data pre-processing operations and corresponding (combination of) basic operators

5. Towards Why+provenance

Given the classification just presented, we can frame the general provenance granularity problem in terms of the two orthogonal dimensions of data derivation, from dataset to item-level, and detail of processor behaviour, from class to internal logic (Fig. 3). When the processors are described using the classification just given, this results in the examples provenance assertions as in Sec. 3. The Figure summarises these for the transformation and selection processors in our use cases, respectively. Representations become more challenging in the lower bottom of the Figure, where we aim to represent processor logic.

When operating at the dataset level, using ActiveClean as an example, the provenance would need to describe processor logic as consisting of three components: “assessment” (A), “cleaning” (C), and “training” (T), and assert that input dataset D is assessed using A, generating a list of data cleaning targets, which is used by C, producing a new version D' of D. D' is used by T to train a model M, which in turn is again used by A in conjunction with D' in the next iteration



Data detail / Processor detail	dataset	→	item
class level			
- transformation	$D \rightarrow D'$		$\{x \rightarrow x'\}_{x \in D, x' \in D'}$
- selection	$D \rightarrow D' \subseteq D$		$\{x \in D \mid \sigma(x') = \text{True}\}$
↓			
logic level	Processor logic		Why x? (transformation, selection) Why x'? (transformation, augmentation)

Figure 3: Data and Processor details and corresponding provenance

of incremental cleaning. Note that PROV is able to support these relationships, after providing the required breakdown of the whole strategy into processes and iterations (Fig. 2(d)).

Supporting the actual *why* questions at item level remains challenging, however. This is the bottom right quadrant in the Figure, where for each $x \in D$, we ask "why did the assessor A choose x for cleaning?" and "how did the cleaner C choose the replacement value?"

The corresponding *why* questions for the selection processes are similar, namely "why did $x \in D$ get selected for removal from the training set?". Note that here the full explanation may be quite involved, as the processor logic involves learning an embedding for D, then clustering in the embedded space, and finally choosing data points based on their distance from the cluster centroids.

While the problem of automatically generating suitable provenance at this level is not fully addressed, it seems that two elements are needed. Firstly, a vocabulary and language, or perhaps a small knowledge graph if relationships are included, to be able to express the concepts mentioned in the provenance narrative above. Such a vocabulary would include a choice of abstraction level, grounded in the baseline classification described in Sec. 4. Secondly, a mechanism to generate provenance assertions that involves *active participation* from the processors themselves, as simply observing the process execution from the outside would not be enough.

6. Conclusions

In this position paper we started from the observation that Data processing workflows for Data Science applications now include sophisticated processors, whose operations are often interleaved with model training. We have suggested increasingly detailed levels of provenance representation, aimed not only at recording granular data derivations, but also to explain *why* each of these derivations occurred. At the deepest level, this may require processors that actively interact with the provenance subsystem during execution, providing the necessary details.

Experimenting with provenance capture models at this level is work in progress. Importantly, we believe this research should be driven by user studies to determine, for different stakeholders, what kinds of explanations are actually expected and desirable.

References

[1] P. Buneman, S. Khanna, T. Wang-Chiew, Why and where: A characterization of data provenance, in: J. Van den Bussche, V. Vianu (Eds.), Database Theory — ICDT 2001, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001, pp. 316–330.

[2] J. Cheney, L. Chiticariu, W.-C. Tan, Provenance in databases: Why, how, and where, Foundations and Trends® in Databases 1 (2009) 379–474. URL: http://dx.doi.org/10.1561/1900000006. doi:10.1561/1900000006.

[3] T. J. Green, G. Karvounarakis, V. Tannen, Provenance semirings, in: Proceedings of the Twenty-Sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS '07, Association for Computing Machinery, New York, NY, USA, 2007, p. 31–40. URL: https://doi.org/10.1145/1265530.1265535. doi:10.1145/1265530.1265535.

[4] Y. Amsterdamer, D. Deutch, V. Tannen, Provenance for aggregate queries, in: Proceedings of the 30th ACM SIGMOD Symposium on Principles of Database Systems, PODS '11, Association for Computing Machinery, New York, NY, USA, 2011, p. 153–164. URL: https://doi.org/10.1145/1989284.1989302. doi:10.1145/1989284.1989302.

[5] W. Oliveira, P. Missier, K. Ocaña, D. de Oliveira, V. Braganholo, Analyzing provenance across heterogeneous provenance graphs, in: Procs. 6th International Provenance and Annotation Workshop, IPAW 2016, McLean, VA, USA, volume 9672, Springer, 2016, pp. 57–70. doi:10.1007/978-3-319-40593-3_5.

[6] A. Chapman, H. V. Jagadish, Understanding provenance black boxes, Distributed and Parallel Databases 27 (2010) 139–167. URL: https://doi.org/10.1007/s10619-009-7058-3. doi:10.1007/s10619-009-7058-3.

[7] M. Stamatogiannakis, E. Athanasopoulos, H. Bos, P. Groth, Prov2r: Practical provenance analysis of unstructured processes, ACM Trans. Internet Technol. 17 (2017). URL: https://doi.org/10.1145/3062176. doi:10.1145/3062176.

[8] Y. Amsterdamer, S. B. Davidson, D. Deutch, T. Milo, J. Stoyanovich, V. Tannen, Putting lipstick on pig: Enabling database-style workflow provenance, Proceedings of the VLDB Endowment 5 (2011).

[9] A. Chapman, L. Lauro, P. Missier, R. Torlone, Supporting better insights of data science pipelines with fine-grained provenance, ACM Trans. Database Syst. (2024). URL: https://doi.org/10.1145/3644385. doi:10.1145/3644385, just Accepted.

[10] S. Krishnan, J. Wang, E. Wu, M. J. Franklin, K. Goldberg, Activeclean: interactive data cleaning for statistical modeling, Proc. VLDB Endow. 9 (2016) 948–959. URL: https://doi.org/10.14778/2994509.2994514. doi:10.14778/2994509.2994514.

[11] F. Neutatz, B. Chen, Z. Abedjan, E. Wu, From Cleaning before ML to Cleaning for ML., IEEE Data Eng. Bull. 44 (2021) 24–41.

[12] S. Krishnan, J. Wang, E. Wu, M. J. Franklin, K. Goldberg, ActiveClean: interactive data cleaning for statistical modeling, Proceedings of the VLDB Endowment 9 (2016) 948–959. URL: https://dl.acm.org/doi/10.14778/2994509.2994514. doi:10.14778/2994509.2994514.

[13] M. Mazumder, C. Banbury, X. Yao, B. Karlaš, e. a. Rojas, DataPerf: Benchmarks for Data-Centric AI Development, 2023. URL: http://arxiv.org/abs/2207.10062. doi:10.48550/arXiv.2207.10062, arXiv:2207.10062 [cs].

[14] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford,

J. Wu, D. Amodei, Scaling Laws for Neural Language Models, 2020. URL: http://arxiv.org/abs/2001.08361. doi:10.48550/arXiv.2001.08361, arXiv:2001.08361 [cs, stat].

[15] A. Abbas, K. Tirumala, D. Simig, S. Ganguli, A. S. Morcos, SemDeDup: Data-efficient learning at web-scale through semantic deduplication, 2023. URL: http://arxiv.org/abs/2303.09540, arXiv:2303.09540 [cs].

[16] B. Sorscher, R. Geirhos, S. Shekhar, S. Ganguli, A. Morcos, Beyond neural scaling laws: beating power law scaling via data pruning, Advances in Neural Information Processing Systems 35 (2022) 19523–19536. URL: https://proceedings.neurips.cc/paper_files/paper/2022/hash/7b75da9b61eda40fa35453ee5d077df6-Abstract-Conference.html.

[17] L. Moreau, P. Missier, K. Belhajjame, R. B'Far, J. Cheney, S. Coppens, S. Cresswell, Y. Gil, P. Groth, G. Klyne, T. Lebo, J. McCusker, S. Miles, J. Myers, S. Sahoo, C. Tilmes, L. Moreau, P. Missier, PROV-DM: The PROV Data Model, Technical Report, World Wide Web Consortium, 2012. URL: http://www.w3.org/TR/prov-dm/.

[18] A. Chapman, P. Missier, L. Lauro, R. Torlone, DPDS: Assisting Data Science with Data Provenance, PVLDB 15 (2022) 3614 – 3617. URL: https://vldb.org/pvldb/vol15/p3614-torlone.pdf. doi:10.14778/3554821.3554857.