Response to Reviewer Comments

Explaining Dataset Changes for Semantic Data Versioning with `Explain-Da-V`

Thank you very much for the opportunity to provide a revised submission of our manuscript. We thank the reviewers for their careful examination of the manuscript and the meta-reviewer for the clear summary of the required modifications. The comments we received were very helpful in revising the manuscript and we believe that responding to the concerns has yielded a much improved version. We fixed all typos and minor comments, given by the reviewers. Changes to the paper are colored in blue. Next, we describe the main changes in the paper according to the stated revision items, addressing the collective concerns of the reviewers:

Revision item 1: *"better clarity on the assumptions, and the criteria that specify when this method is applicable"*
Related Reviews:

R1W1: *"The paper is unclear about its assumptions, and the criteria that specify that this method is applicable (e.g., is there a restriction on "how far apart" the versions are?)"*

R3O1: *"My overall point of weakness in the paper is that it presents a bit like a laundry list of solutions without grounding. I am missing the paragraph that says "we looked at all these versions of tables and distilled the possible core changes to be XYZ". The paper gives a lot of cases and derived features, and I'm unclear what is really necessary and if there is part of the space that is not covered."*

Response 1: The main assumptions of this paper are:

- **Table versions are given with an alignment between them**: The focus of this paper is to explain a set of changes between a pair of versions. We assume we are given two tables where one is known to have been derived from the other and we know a match between the attributes and tuples that the two tables share. While interesting, we are not considering the version search problem where we try to find tables that are versions of each other.
- **Changes are internal**: In this paper we focus on internal changes only. In the submission, we had mentioned this in Section 5.1 and now have move this up to the end of the introduction for clarity. As we state in the conclusion of the paper, in future work we intend to "address changes that are triggered by external data such as performing joins and unions."

In addition to these main assumptions, we follow recent work [74] and design a data-driven approach as "metadata is not always complete and may be ambiguous" (Section 2.1), "reflect[ing] that we only use data values (we do not use meta-data such as headers)" (Section 3.2). To make these assumptions clearer, we added the following paragraph at the end of the introduction: "In this paper, we assume we are given two tables where one is known to have been derived from the other (i.e., is a version it) and we know a match between the attributes and tuples that the two tables share. This work focuses on "internal" additions, deletions, or modifications (modeled as deletions followed by additions). External additions, for example, finding joinable tables [103] and joining them with a table to create a new version, are reserved for future work."

Aiming at better clarity in the paper, its assumptions and applicability, we followed the suggestion of R3 (O2) and changed the structure of the paper. Specifically, the core semantic explanation methods are now introduced in a separate section (Section 4), which also highlights the criteria on which a method is applicable. Specifically, referring to R1W1, we clarified that our method can be applied regardless of how far apart the versions are ("Accordingly, `Explain-Da-V` can be applied over any pair of versions...", Section 4). The new structure permitted us to add the requested paragraph (first paragraph of Section 4) that describes how we distilled the set of possible changes we consider. The methods are then used in Section 5-6. Accordingly, Section 5.1 was updated to better reflect when and how the different methods are used in practice using the running example. We also created a pseudo code for the algorithm that explains attribute additions (Section 5.1), which, due to page limit, is provided in a technical report [88].

Revision item 2: *"better view on which transformations can be supported/explained in the proposed system"*
Related Reviews:

R2W1(+D1): *"The scope of exactly what transformations can be supported/explained in the proposed system is missing, as many transformations would not be explained using the proposed techniques."*

Response 2: The new structure (see Revision item 1) has helped us to better scope the transformations supported (and not supported) by `Explain-Da-V`:

- **Numeric Transformations**: "Note that while these extensions...", Section 4.1.
- **Categorical Transformations**: "Note that decision trees cover only explanations...". Section 4.2
- **Textual Transformations**: last paragraph Section 4.3.

- **Reshaping Transformations**: "Currently, Explain-Da-V does not support...", Section 4.5.

In addition to the beginning of Section 4, the specific association between the types of origin and goal relations and the method is now clearly highlighted (in bold) throughout Section 5. We believe the new organization allows us to make more precise statements on scoping.

Revision item 3: *"more details on the experimental evaluations"*
Related Reviews:

R2W4 (+D2): *"Details on experimental evaluations can be improved."*

R3O3: *"For experiments, I'd also like to see or described the number of transformations/explanations returned. I imagine there should be some tradeoff between validity and generality. The paper would also be stronger if there was one other dataset added. I really like your benchmark datasets but the only dataset you have where the lifts with your methods aren't as massive is the already existing one where you can't measure generality. I worry a bit that the benchmark is too "fit" to your method with the results as they are."*

Response 3: We updated the empirical evaluation section (Section 8) to provide additional details. Specifically, we

- updated the description of the new benchmark and its generation ("We design a new benchmark for the novel task of...", Section 8.1.1),
- added an "Evaluation Measures" section to clearly state the measures we use in experimental evaluations ("The explanations provided by our baselines are of a single type...", Section 8.1.4) including two additional measures following the reviewers comments, namely the proportion of 100% valid/generalizable explanations returned by Explain-Da-V and the average number of explanations (# $\mathcal{E}$) from which the method selects the most explainable valid method,
- extended Table 3 to report these measures and updated the discussion in Section 8.2.
- generated an extension of Table 3 which provides results by transformation types following R2 (D2) suggestion. The new analysis highlights the dominance of Explain-Da-V when it comes to numeric and categorical transformations with very significant improvements. Moreover, this analysis also shows that Explain-Da-V out-performs (+51% and +33% average improvement of validity and generalizability, respectively) the baselines also for the specific case of textual transformations, which is the main focus of the reported baselines. Due to space limitation, this table is provided in a technical report [88].

We are confident that these additions to the empirical evaluation not only provide additional details but also emphasize that Explain-Da-V covers a wide variety of cases including improving prior art (e.g., textual transformations) and solving new existing challenges (e.g., numeric transformations). We also note that the Auto-Pipeline Benchmark and our new benchmark (SDVB) contain a variety of scenarios, ranging in domain, involving different change dimensions (or pipelines in Auto-Pipeline Benchmark), and requiring different methods. When designing the new benchmark, we followed common data preparation pipelines, some of which do not fit the current state of Explain-Da-V. One example of such a transformation is given in Section 8.2 ("For example, one of the IMDB version-sets involves..."). Other examples of challenges introduced in the benchmarks are small tables (less samples for a model to learn from), compound textual transformations (the models timeout), non-exact tuple de-duplication, idiopathic changes (randomly removing/adding tuples), and others.

Revision item 4: *"discussion on metrics choices"*
Related Reviews:

R1W3: *"I don't fully understand the metrics the paper uses, in particular generalizability."*

Response 4: We updated Section 7.1 to better reflect the importance of validity and generalizability. Specifically, we clearly highlighted the limitation of solely looking at validity and the need for generalizability ("As illustrated in the example, validity only looks at the given dataset...", Section 7.1). Moreover, to better explain the generalizability measure, we introduced an additional example introducing "two dataset versions. The top table is similar to Fig. 1a and the bottom table corresponds to Fig. 5 such that the same transformations over Fig. 1a to generate Fig. 5 were applied here as well. (Example 13). This example illustrates a scenario where two explanations are equivalent in their validity, but differ in their generalizability.

Revision item 5: *"an improved problem definition"*
Related Reviews:

R1W2: *"The main problem definition appears simplistic and not as well-thought out as other parts of the paper."*

Response 5: We introduced a new problem definition (Section 7.2) that uses the validity measure to define the type of explanations we aim to find. Specifically, Section 7.1 defines "What makes an explanation better than another?" and is then used in the problem definition. We also clearly state that we "aim to generate "explanable" explanation, that is, an explanations that a user can understand." (Section 7.2), which is then discussed in Section 7.3. Finally, Section 7.4 highlights that we do not only search for valid explanations that satisfy Definition 14, we aim to find "the most explainable valid solution".

Other comments:
R2W2: *"The paper describes a system that integrates PBE, QBO, decision tree, and regression models together, for the application of change explanation. These pieces are glued together using ad-hoc definitions of conciseness/concentration, etc. Overall, the algorithmic contribution of the paper is not particularly strong."*
→ We have reorganized the paper to highlight the core semantic explanation methods that we use and why (new Section 4). We believe this clarifies the systematic, rather than *ad hoc*, choices made in defining the explanation methods. It is correct that our contributions (as outlined in the end of Section 1) are not algorithmic.

R2W3: *" It is unclear whether an inferred transformation that is correct on 70-80% of values in a column (measured by validity or generalizability in the paper) is really good enough in the proposed application that helps users to understand the underlying transformation."*
→ First, we truly believe that a user can benefit from a explanation (transformation) that is only 70-80% correct compared to no explanation at all. For example, recall the transformation over a3 to create a7 in our running example (Fig. 2). The following explanation $\left(a3, \begin{cases} 4, & \text{if } 8.5 \le a3 \\ 3, & \text{if } 8 \le a3 < 8.5 \\ 2, & \text{if } 7.5 \le a3 < 8 \\ 1, & \text{otherwise} \end{cases}\right)$ would only be 75% valid (it will incorrectly generate the value 4 for tuple m2); yet, beneficial for the user to understand, for example, that a3 was discretized to create four rating classes. In the revised submission we also measure the proportion of 100% valid/generalizable explanations returned by Explain-Da-V (see Revision Item 3). Using this measure, we show that even if a user is interested only in 100% valid/generalizable explanations, Explain-Da-V still returns much more valid/generalizable explanations than the baselines (see Table 3 and discussion in Section 8.2 - "Even when we evaluate only the 100% valid/generalizable…").

R1W4: *"The metrics and evaluation do not appear to align strongly with the paper's motivation. (E.g., it is not clear how they capture the semantics of changes, which is at the core of the motivation for this work.)"*
→ As we describe in the introduction, the changer's intent, which is subjective, cannot be truly reverse engineered. Thus, our objective is to provide the user of Explain-Da-V an accurate explanation, e.g., a set of functions, that describes the changes. Specifically, if an explanation captures the semantics of change, it means it can reproduce the change and generalize it beyond a single specific setup. We updated the opening of Section 7 accordingly.

R1D2: *"An another example, the framework models additions and removals, but it is not clear if/how it handles modifications. Do these need to be modeled with a combination of removal and addition transformations?"*
→ Yes. We clarified this in Section 3 - "Modifications can be modeled as a removal followed by an addition."

R1D5: *"The solution is heuristic, although the various components seem reasonable and they are mostly intuitive. However, I would like more discussion on some of the choices, e.g., why do we need feature extensions, is the increased search space reasonable, etc."*
→ As described above, we have reorganized the paper, which allows us to clarify the systematic, rather than ad hoc, choices made in defining the explanation methods. Specifically, we (1) emphasized that the feature extensions are motivated by commonly used data science and engineering operations [64] (Section 4.1) and (2) explained the importance in the increased search space ("Note that Foofah aims to synthesize transformations…").

R2W5: *"Explanation for row-wise removal/addition (Section 5) seems very limited (e.g., only NaN and duplicates are considered, plus QBO based integrations)."*
→ Among change explanations, our main contribution is in vertical additions (adding attributes), which follows and extends a significant body of work on data transformation by example (see Section 2.3). Row-wise changes are more limited by nature and were explored as well in the literature by, for example, SQUARES [77] and AutoPandas [18]. In contrast, Auto-pipeline does "not consider row-level filtering" [100]. Both SQUARES and AutoPandas follow the programming-by-example paradigm (see Section 2.3) and explore row-wise transformations as a part of their search space. For example, SQUARES aims to find "filters" while AutoPandas search space contains (the non-parameterized) `drop_duplicates` `dropna` operators (see https://rbavishi.github.io/autopandas/#autopandas-supported-functions). Explain-Da-V covers both and others (e.g., outlier detection). Also the way we address selection predicates ("filters" in SQUARES) is much more salable using the categorical change explanations (see Section 4.2 and Section 6.1). Finally, note that in our experiments we mention that Explain-Da-V outperformed SQUARES even when considering only horizontal removal explanations ("Despite its focus on learning…", Section 8.2).

R3D1: *"In related work, how is this related to query explanation work."*
→ We thank the reviewer for the reference. We assume that the comment referred to the work on explaining query answers [84]. This line of work different that ours in setup and explanation methods. They are given a query and a database, and explain interesting or unexpected query answers using the tuples in the given database. Ours, on the other hand, explain dataset changes using data transformations. While

changes can be thought of as a set of queries, the explanations for queries that involve non-trivial transformations (e.g., adding the column a5 in the running example) cannot be explained using the tuples themselves. We update the related work section accordingly ("Finally, another related research is in explaining query answers…").

R3D2: *"In preliminaries, is r_im the right notation? Do you mean r_in?"*
→ Yes. Thanks for the clarification. The mistake was corrected in the revised version.

R3D4: *"Do you handle external joins that add new columns?"*
→ No. In this paper we do not address external changes such as joins and unions. We improved the clarity of assumptions (and moved them to introduction), please see Revision item 1 for additional details.

R3D5: *"In 4.1.4 do you use the same derived feature set as above? The set with the norms, and exps, and math operators,…"*
→ No. It the current form of Explain-Da-V, we do not include the numeric extensions (the norms, and exps, and math operators,…) in categorical explanations (Section 4.2). We thank the reviewer for the comment and will definitely consider using extensions for categorical explanations in future work.

R3D6: *"I got a bit confused over 5.1. Why are you searching for duplicate tuples. And in example 12, why are you adding one hot encoded features. Where are the functions for encoding coming from?"*
→ Section 5.1 in the previous submission (Section 6.1 in the revised submission) aims to explain removed tuples. A common reason users remove tuples is deduplication (aka entity resolution/matching). Accordingly, Explain-Da-V searches for duplicate tuples as a possible explanation. For example, assume that a tuple r_i appears in T but not in T' (i.e., $r_i \in$LΔ_r).Explain-Da-V$aimstofindatupler'_j \in T'$ that is a duplicate of r_i. If such a tuple is found, Explain-Da-V provides an explanation to the user that r_i was removed because it was duplicated. We made this clearer in the revised paper, please see second paragraph of Section 6.1 ("For table-dependent explanations, we aim…").

Example 12 in the previous submission (Example 11 in the revised submission) uses a categorical change explanation method (Section 4.2) to find a joint explanation. Since the example includes mixed types, the decision tree is applied also over encoded (using categorical-encoding change explanation, Section 4.4) attributes in the table (please see the paragraph preceding this example is the revised submission).

Finally, we updated Section 4.4 to better describe the generation process and the importance of the encoding.

RS: remember to add a reminder to the technical report. We are confident that these changes improved the paper and would again like to thank the reviewers and the meta-reviewer for their input.

Best regards,
Roee and Renée.

Explaining Dataset Changes for Semantic Data Versioning with Explain-Da-V

Roee Shraga, Renée J. Miller
Northeastern University
Boston, MA, USA
{r.shraga,miller}@northeastern.edu

ABSTRACT

In multi-user environments in which data science and analysis is collaborative, multiple versions of the same datasets are generated. While managing and storing data versions has received some attention in the research literature, the semantic nature of such changes has remained under-explored. In this work, we introduce Explain-Da-V, a framework aiming to explain changes between two given dataset versions. Explain-Da-V generates *explanations* that use *data transformations* to explain changes. We further introduce a set of measures that evaluate the validity, generalizability, and explainability of these explanations. We empirically show, using an adapted existing benchmark and a newly created benchmark, that Explain-Da-V generates better explanations than existing data transformation synthesis methods.

PVLDB Reference Format:
Roee Shraga, Renée J. Miller. Explaining Dataset Changes for Semantic Data Versioning with Explain-Da-V. PVLDB, (): XXX-XXX, .
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/shraga89/ExplainDaV.

1 INTRODUCTION

Data is one of the most important ingredients in any decision making process. The amount and size of data is growing and datasets are being reused for multiple analyses. Data may be stored in different systems (e.g., data lakes [74]), vary in their formats, and may or may not contain metadata. Data projects often involve multiple users that work on datasets conjointly or independently, creating different data versions. Accordingly, *data versioning* becomes an important ingredient in data management [20]. Nevertheless, even if versions are well managed [21], the documentation may be superficial, e.g., embedded in filenames, which can be very inadequate. In addition, the collaboration itself may not be structured or properly managed and each user may perform different, often *undocumented* processing steps on data [54, 59–61, 101]. For example, some users may clean the data by removing rows or columns if they have duplicated or missing information. Other users extract features, transforming the current data to create new columns.

Proceedings of the VLDB Endowment, Vol. , No. ISSN 2150-8097.
doi:XX.XX/XXX.XX

Current tools have limited data versioning support [59]. Generally speaking, data, as opposed to code, may be less documented [60, 101] and data changes, even if documented, are rarely accompanied by useful descriptions, making it difficult to understand them [54]. Within a close collaboration group, a notebook containing transformation code may be shared, but between organizations this is rarely done. Consider, for example, the many versions of important datasets shared on open data portals [41, 42, 55] where transformations are generally not shared. The lack of sufficient version documentation results in reduced reproducibility and trust among users using the data [61, 101]. While managing and storing data versions has received attention in literature [20, 21, 56, 85], the semantic nature of such changes has remained under-explored. We motivate our work using the following example.

a0	a1	a2	a3	a4
m1	The Godfather (A)	175	9.2	Drama
m2	Hamilton (PG-13)	160	8.6	Drama
m3	The Avengers (UA)	143	8.0	Action
m4	Inception (UA)	NaN	8.8	Action
m5	Moana (U)	107	7.6	Animation

(a) Dataset version created by UserA

a0	a1	a2	a3	a4	a5	a6	a7	a8
m1	The Godfather (A)	175	9.2	Drama	A	2.91	4	17
m2	Hamilton (PG-13)	160	8.6	Drama	PG-13	2.67	3	16
m3	The Avengers (UA)	143	8.0	Action	UA	2.38	3	17
m5	Moana (U)	107	7.6	Animation	U	1.78	2	9

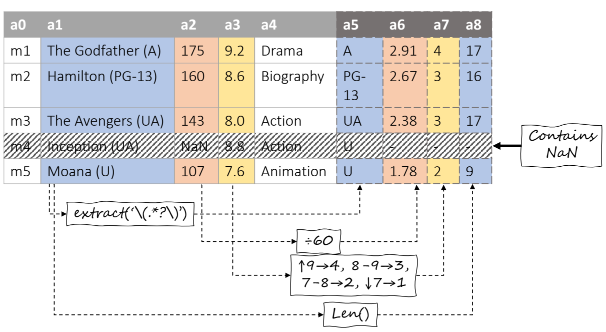
(b) Dataset version created by UserB

Figure 1: Example dataset versions about movies created by two users. Attribute names are provided in Example 1.

Example 1. *Figure 1 presents two dataset versions about movies. We discard the column names from the figure to illustrate a realistic (data lake) scenario. For readability,* a0 *is a tuple id,* a1 *represents the movie title,* a2 *measures the movie runtime in minutes,* a3 *assigns a rating to the movie, and* a4 *provides the genre of the movie. For convenience of presentation, lets assume that the table on the bottom (Figure 1b) was created by UserB as a derivation of the table on the top (Figure 1a) that was created by UserA. Even properly naming the tables, e.g., Table 1a as* data1_v1.csv *and Table 1b as* data1_v2.csv, *or knowing that Table 1b is derived from Table 1a [20], does not help UserA to get a* semantic *understanding of what UserB changed in the table or, more importantly, what data processing steps UserB has performed.*

Example 1 illustrates the need for a semantic understanding of a new dataset version. Aiming to fill this gap, this work provides the setup and new solution to *explain* the semantic changes between two dataset versions. Specifically, our goal is to *automatically* explain (in a simple user friendly way) the steps leading from one version of dataset to the other. For example, *how was column* a6 *in Figure 1b created?* or *why was the fourth row in Figure 1a deleted?* Note that the changer's intent, which is subjective, cannot be truly reverse engineered. Our objective is to provide the other user an accurate explanation, e.g., a set of functions, that describes the changes. Following this goal, we return to our motivating example.

Example 1 (cont.). *Figure 2 illustrates an annotated version of Figure 1b, that explains the changes. In other words, Figure 2 reverse engineers the changes made by UserB in a way that a user can understand. Specifically, UserB cleaned the rows that contain* NaN *values (in this case* m4*) and extracted numerical features. The certification of the movie, given in parenthesis in* a1*, was extracted to create* a5 *and the column* a6 *converts the units of* a2*, the runtime of the movie, from minutes to hours. Since the range of movie ratings (*a3*) is limited, UserB also discretized the values to create four rating classes in column* a7*. Aiming to examine the effect of title length (an effect found for paper citations [43]) within the domain of movies, UserB added column* a8 *that provides the length of titles from* a1.



a0	a1	a2	a3	a4	a5	a6	a7	a8
m1	The Godfather (A)	175	9.2	Drama	A	2.91	4	17
m2	Hamilton (PG-13)	160	8.6	Biography	PG-13	2.67	3	16
m3	The Avengers (UA)	143	8.0	Action	UA	2.38	3	17
~~m4~~	~~Inception (UA)~~	~~NaN~~	~~8.8~~	~~Action~~	~~U~~	-	-	-
m5	Moana (U)	107	7.6	Animation	U	1.78	2	9

Figure 2: An interpretation of the changes between the dataset versions given in Figure 1. The columns are colored based on their origin (e.g., a5 is blue because it originates from the blue a1) and annotated column transformations are given at the bottom. The annotated row transformation is given on the right, in this case removing a row, which is also illustrated by diagonal stripes over the row.

As illustrated in Example 1, there are a variety of possible transformations (e.g., multiplying/dividing the values of a numeric column, e.g., a3, by a constant), potentially creating an infinite possible number of changes to a dataset. These changes can be vertical (changing columns) or horizontal (changing rows), they can add information (adding columns/rows) or remove information (removing columns/rows) and they can involve different data types, e.g., textual to numeric (a1 to a8) or numeric to categorical (a3 to a7). In addition, a user may also change a cell in the table, e.g., replacing the NaN value in row m4 by 146.25 (the mean value of the other values in the column), or perform a full-table operation, e.g., transposing the table. Transformation discovery methods are used for multiple data management tasks including fuzzy joins [104], data wrangling [24], entity consolidation [44] and more [17, 51, 53, 57] mainly focusing on textual (text-to-text) transformations and consider the transformed values (rather than the transformation itself).

Our method mainly focuses on data versioning, for which, the transformations themselves, as a means of explaining changes among different dataset versions, is the main interest. Our resolved transformations also cover transformations that involve, among others, numeric transformations. The term "explanation" became quite common recently and may be associated with multiple meanings. For example, both El Gebaly et al. [46] and Kim et al. [62] use data summaries as explanations. Explain3D [97], which shares a similar context to ours, explains dataset disagreements with syntactic provenance-based and value-based modification mappings. In this work the main component of an explanation is a transformation that explains change. This paper makes the following contributions.

(1) **Semantic Data Versioning Definition:** we define and solve a novel problem of semantic data versioning by explaining the changes between two dataset versions.

(2) **Vertical and Horizontal Data Transformation Resolution Across Different Data Types:** we present a solution to the problem of semantic data versioning that examines both vertical (adding/removing columns) and horizontal (adding/removing rows) transformations that involve multiple data types.

(3) **Semantic Data Versioning Metrics:** we provide a set of evaluation measures to examine the quality of explanations in terms of validity, generalizablity, and explainability.

(4) **Semantic Data Versioning Benchmark:** we introduce a new data versioning benchmark composed of 5 version-sets including 342 different dataset versions representing a total of 1702 changes.[1]

(5) **Empirical Evaluation:** our experiments show that Explain-Da-V performs better than multiple baselines on both our new version benchmark and on an existing data science pipeline benchmark [1]. We analyze the impact of different components of our solution on performance.

In this paper, we assume two tables are given where one is known to have been derived from the other (i.e., is a version it) and we know a match between the attributes and tuples the two tables share. This work focuses on "internal" additions, deletions, or modifications (modeled as deletions followed by additions). External additions, e.g., finding joinable tables [103] and joining them with a table to create a new version, are reserved for future work.

2 RELATED WORK

We are, to the best of our knowledge, the first to address the semantic aspect of data versioning. Yet, related research exists ranging from synthesizing data transformations to exploring data change.

2.1 Data Versioning

Data versioning research mainly focus on developing version managers to decrease the need for storing many versions of large datasets [74]. For example, DataHub provides a git-like interface to manage, store, recreate, and retrieve versions using a directed version graph [20]. Follow-up research further studied the trade-off between recreation and storage in a principled way analyzing six different settings [21]. Recently, Schüle et al. presented TardisDB [85], an SQL extension to support version management. TardisDB uses named branches over tables, to monitor table versions and track

[1]We will make code and benchmark publicly available upon acceptance. Main parts of the code and a benchmark sample are available [15].

their modification history. In contrast, we focus on the semantic aspects of data versioning, zooming in on explaining the semantic differences between dataset versions. Schema versioning has also been studied [82]. Although schemata may change over time [91], which provides semantic hints to data change, we assume metadata is not always complete and may be ambiguous [74]. Hence, we focus only on the versioning of the data itself.

Multiple methods find or discover related tables [32, 40] (e.g., joinable [103] and unionable [75]). While different versions of a table may be related and found using these methods, our work assumes that the discovery has already been done and aims at providing a semantic explanation for the differences between versions.

2.2 Data Change, Difference, and Integration

We assume that some match between the attributes and tuples of the versions is given. This assumption is rooted in many years of data integration research, exploring attribute matching (schema matching) [81, 87], tuple matching (entity resolution) [47, 67], and others [19, 70]. Earlier works looked into change and copy detection in structured data [33, 34], which was later extended also to semi-structured documents such as XML [37, 76, 98].

Acknowledging data change, Bleifuß et al. [22] envision systems that can interactively explore such change. They present a model of what changed, where, when and how, using what they call a change-cube to monitor the history of changes over time using methodologies such as time-series clustering [25]. DBChEx [23] is a tool to explore data and schema change using a set of exploration primitives. While similar in nature, this line of work focuses mainly on how to *explore* change aiming to answer questions such as "How many changes have there been in recent minutes?" and "How old are the entities in table Y? When were they last updated?" [22, 23]. Our work focuses on local changes between versions and *how* the changes were performed (which transformations were applied?), e.g., how did UserB create the table in Figure 1b from Figure 1a.

Finally, another related research is explaining query answers [69, 84]. Given a query and a database, they explain query answers using the tuples in the given database, e.g., using provenance [35]. We, contrarily, explain dataset changes using data transformations. While changes can be thought of as a set of queries, explanations for queries that involve non-trivial transformations (e.g., adding a5 in Example 1) cannot be explained only using the tuples.

2.3 Data Transformation By Example

The final related line of research we cover aims to automatically transform data. Largely, given input and output tables (datasets) or their subsets (examples), the goal of such approaches is to find a transformation (program) such that if it is applied over the input we get the output. It is worth noting that earlier work has referred to this problem as query reverse engineering [77, 95], which is roughly the same idea, i.e., finding a query that generates the output using the input. This line of work can be divided into two main groups.

The first group is rooted in a paradigm called programming-by-example (PBE) [24, 49, 51, 57, 58, 89, 90, 104], where the goal is to synthesize a program that manipulates a given input to get a given output. To do so, methods design different search spaces (operators to be applied over the input) and apply different search algorithms. For example, Foofah [57] creates a search space using operators such as drop (delete a column) and split (separate a column by some delimiter) and search the space using A* heuristic search. Clx [58] also introduces string patterns such as regular expressions to the search space and tokenizations. Data Diff [92] applies a search approach to "patch" transformations, summarizing distribution changes, including one numeric patch (operator) supporting linear transformations with pre-defined (randomly selected) parameters. Muller et al. describes differences between relational databases with what they call "update distance" [72] using a similar searching approach. Finally, Bogatu et al. introduced functional dependencies to navigate the search space [24], which we also use in our work.

The second group focuses on creating transformation repositories from external sources such as Web Forms, Knowledge Bases [17, 78], GitHub and Stackoverflow [52, 53]. Transform Data by Example (TDE) [52], instead of searching through a space of predefined possible operations, creates a search engine where transformation functions are crawled from GitHub and Stackoverflow. Instead of applying heuristic search such as A*, TDE ranks candidate functions to find relevant functions. TDE was later extended to allow transformation search based on patterns [53]. DataXFormer [17] and Proteus [78] create a repository of tables from which desired output values can be extracted.

Asimilar line of research revolves around resolving data preparation and analysis transformations [18, 99, 100]. AutoPandas [18] focuses on Pandas library [14] and aims to synthesize a program using pandas functions. Auto-pipeline [100] extends the "by-example" paradigm to "by-target", meaning that the output the user provides is not necessarily aligned with the input and can require table-reshaping operations (e.g., group by). Auto-pipeline comes in two variations, namely, search (which is equivalent, yet extended to what is described above) and deep reinforcement learning. The former can be a candidate baseline for our approach.[2] The latter requires training data which we assume does not exist in our setting.

In contrast to PBE and query reverse engineering, we do not focus on matching input to output. Rather than looking only at success rates (is the transformation valid), our search is guided by the principle of creating valid, generalizable, and explainable transformations. In our approach, these transformations can be multi-dimensional (adding/removing attributes/tuples) and address multiple data-types (e.g., numeric, categorial, and text transformations). While our string-based transformation resolution is based on an extended Foofah, we also support numeric transformations using explainable machine-learning algorithms to *fit the appropriate transformation* rather than searching a very large space of possible transformations. We go beyond Foofah to support text-to-numeric transformations (e.g., measuring the length of a string) and text cleaning operations (e.g., stopword removal and lemmatization).

3 SEMANTIC DATA VERSIONING

A dataset is denoted by a table T, composed of a set of attributes $T_A = \{A_1, \dots, A_n\}$ and tuples $T_r = \{r_1, \dots, r_m\}$. Each tuple is defined as $r_i = \langle r_{i0}, r_{i1}, \dots, r_{in} \rangle$, such that r_{i0} is the tuple identifier and r_{ij} ($j \neq 0$) is a value assigned to the attribute A_j in the tuple r_i.

[2] Reproducible code of Auto-pipeline is not publicaly available, so, in our experiments, we reproduced its functionality using Foofah to allow a quantitative comparison.

Table 1: Notations used in the paper. The changes $L\Delta_A$, $L\nabla_A$, $L\Delta_r$, and $L\nabla_r$ are defined wrt the left-hand table T. The right-hand notation can be obtained by replacing T with T' below.

	Notation	Meaning	Notation	Meaning
basic	T	Left-hand dataset	T'	Right-hand (revised) dataset
	T_A	The attribute set of dataset T	T_r	The tuple set of dataset T
changes	$L\Delta_A$	Unmatched attributes in T $\{A_i : A_i \in T_A \wedge \nexists A'_j \in T' : (A_i, A'_j) \in \Sigma_A\}$	$L\nabla_A$	Matched attributes in T $T_A \setminus L\Delta_A$
	$L\Delta_r$	Unmatched tuples in T $\{\pi_{L\nabla_A}[r_j] : r_j \in T_r \wedge \nexists r'_i \in T : r_{0i} = r'_{0i}\}$	$L\nabla_r$	Matched tuples in T $\{\pi_{L\nabla_A}[r_j] : r_j \in T_r\} \setminus L\Delta_r$

Often we may have two datasets and know one was derived from the other but the actual transformation code or documentation has been lost [54, 59–61, 101]. In what follows, we address the problem of explaining the changes between two dataset versions.

Given two dataset versions, T and T', we assume the latter, wlog, is a *derived table*, i.e., a user changed the table T and as a result obtained the table T' with $T'_A = \{A'_1, \ldots, A'_n\}$ and tuples $T'_r = \{r'_1, \ldots, r'_m\}$. We assume that an alignment between T_A and T'_A (attribute-match, denoted Σ_A) is given and that tuples in T and T' with the same identifier (r_{0i} and r'_{0i}) are assumed to represent the same real world entity. An attribute $A_i \in T$ (or $A'_j \in T'$) is considered unmatched if it does not appear in Σ_A. A record $r_i \in T$ (or $r'_j \in T'$) is considered unmatched if there is no record in T' (respectively, T) with the same identifier.

Given an attribute-match Σ_A, we define the changes between the two dataset versions to be explained using a three symbols notation. The first refers to whether the dataset is the left-hand one (L) or the right-hand (revised) one (R), the second to whether it is the matched (∇) or unmatched (unmatched is also called delta (Δ)), and the third refers to attributes (A) or tuples (r). Specifically, $L\Delta_A$(left-hand delta attributes) and $L\nabla_A$(left-hand matched attributes) are the set of unmatched (delta) and matched (consistent) attributes in T, respectively. Similarly, $R\Delta_A$ and $R\nabla_A$ are the unmatched and matched attributes in T'. Using these sets, we create *projected* tuples. Let r_j be a tuple of table T, the projected tuple is given by $\pi_{L\nabla_A}[r_j]$, projecting out non-matching attributes. Given such projected tuples, we can define similar sets for tuples, namely $L\Delta_r$(left-hand delta tuples), $R\Delta_r$(right-hand delta tuples), $L\nabla_r$(left-hand consistent tuples) and $R\nabla_r$(right-hand consistent tuples). We summarize this notation in Table 1. Intuitively, we are interested in explaining the deltas between the datasets, i.e., $L\Delta_A$, $R\Delta_A$, $L\Delta_r$, and $R\Delta_r$.

Example 2. *Given the dataset versions in Figures 1a (T) and 1b (T'), the attribute-match is simply given by aligning the columns headers (e.g.,* `a2` $\leftrightarrow$ `a2`*). The tuple ids are given under* `a0` *(e.g.,* `m1` $\leftrightarrow$ `m1`*). The following are the change sets: $L\Delta_A = \emptyset$ (no removed columns), $R\Delta_A = \{$*`a5`*,* `a6`*,* `a7`*,* `a8`*$\}$ (four added attributes), $L\Delta_r = \{$*`m4`*$\}$ (one removed tuple), and $R\Delta_r = \emptyset$ (no added tuples).*

3.1 Change Explanations

We use the term *explanation* to refer to a user friendly way to interpret a change between two relations. Intuitively, an explanation is a transformation $\mathcal{P}$ from an origin O to a goal $\mathcal{G}$. Formally, an explanation $\mathcal{E}$ is defined with respect to a *goal* $\mathcal{G}$ with a name $\mathcal{G}_{name}$ and an associated relation $\mathcal{G}_{relation}$ it represents. As the goal, the origin is also associated with a name (O_{name}) and a relation ($O_{relation}$). A transformation $\mathcal{P}$ is an expression that transforms the origin relation $O_{relation}$ into the goal relation $\mathcal{G}_{relation}$. The origin relation may also be empty. A formal definition is as follows

Definition 3 (Explanation ($\mathcal{E}$)). *Let $\mathcal{G}$ be a goal. An* explanation *$\mathcal{E}_{\mathcal{G}} = (O, \mathcal{P})$ of $\mathcal{G}$ is composed of an origin O and a transformation $\mathcal{P}$, such that $\mathcal{G}_{relation} = \mathcal{P}(O_{relation})$.*

3.2 Explaining Dataset Changes

We focus on two *orientations* of explanations, namely, *vertical* explanations and *horizontal* explanations. We further distinguish between *removal* and *addition* explanations. Modifications can be modeled as a removal followed by an addition. Explanation types differ in the type of relations that the origin and the goal represent.

The goal ($\mathcal{G}_{relation}$) and origin ($O_{relation}$) relations are defined with respect to versions T or T'. Specifically, the relations $O_{relation}$ and $\mathcal{G}_{relation}$ can be a projection (subset of attributes) or selection (subset of tuples) over either T or T'. In vertical explanations (adding or removing attributes), the associated goal and origin names ($\mathcal{G}_{name}$ and O_{name}) are the projected attributes. For horizontal explanations (adding or removing tuples), $\mathcal{G}_{name}$ and O_{name} correspond to the set of tuple ids in the subset. When clear from context, we refer to the goal and the origin by their names.

Example 4. *Recall the versions in Figures 1a (T) and 1b (T'). An example explanation for a goal $\mathcal{G} = (a6, \pi_{a6}[T'])$ is composed of an origin $O = (a2, \pi_{a2}[T])$ and a transformation $\mathcal{P} = \pi_{a2}[T] \div 60$, which is tuple-based, i.e., divide each tuple in $\pi_{a2}[T]$ by 60. When clear from context, we denote this explanation as $\mathcal{E}_{a6} = (a2, a2 \div 60)$.*

Recalling the change sets that we aim to find vertical addition explanations for $L\Delta_A$, vertical removal explanations for $R\Delta_A$, horizontal addition explanations for $L\Delta_r$, and horizontal removal explanations for $R\Delta_r$. An explicit problem definition, that relies on the quality of explanations is provided in Section 7.2.

Example 5. *Recall the versions of Figure 1, annotated changes in Figure 2, and $L\Delta_A$, $R\Delta_A$, $L\Delta_r$, and $R\Delta_r$, defined in Example 2. A possible set of explanations to explain the changes is as follows*

$$
\begin{aligned}
\mathcal{E}_{\mathbf{a5}} &= (a1, \texttt{extract(a1, '(.*?)')}) \\
\mathcal{E}_{\mathbf{a6}} &= (a2, a2 \div 60) \\
\mathcal{E}_{\mathbf{a7}} &= \left(a3, \begin{cases} 4, & \text{if } 9 \le a3 \\ 3, & \text{if } 8 \le a3 < 9 \\ 2, & \text{if } 7 \le a3 < 8 \\ 1, & \text{otherwise} \end{cases} \right) \\
\mathcal{E}_{\mathbf{a8}} &= (a1, \texttt{len(a1)}) \\
\mathcal{E}_{\mathbf{m4}} &= (\emptyset, \texttt{has\_NaN})
\end{aligned}
$$

Most of the explanations are self-explanatory (as they should be). Interesting cases are $\mathcal{E}_{a5}$, for which we use some programming language notation to express that a value, given in parenthesis, is extracted from the column `a1`*. Another example is the horizontal explanation $\mathcal{E}_{m4}$,for which the origin is an empty set and the tuple was removed due to a* `NaN` *(null equivalent) value. We use* `has_NaN` *for convenience, in practice, the transformation is one that removes tuples with NaN values from T. Although the goal is* `m4`*, the transformation we find is one that removes* `m4` *and no other tuples.*

Explain-Da-V is a data-driven[3] method composed of four parts corresponding to adding/removing attributes/tuples. We first describe our core explanation methods (Section 4), which are then utilized to explain vertical (Section 5) and horizontal changes (Section 6).

4 CORE SEMANTIC EXPLANATION METHODS

Our core explanation methods rely on fitting an appropriate explanation methodology to data types we find in the origin $\mathcal{O}$ and the goal $\mathcal{G}$. Rather than the traditional database attribute types (strings, integers, floats, etc.), given the nature of our analysis, we look into ML feature types [86]. We focus on three main types, namely, *Numeric*, *Categorical* and *Textual* (mixed types are considered textual), which characterizes the core changes Explain-Da-V covers.[4] Aiming to resolve a high variety of changes, we develop methods that are built on top of different types of origin sets using multiple approaches that exploit the type of change. Accordingly, Explain-Da-V can be applied over any pair of versions, regardless of how far apart the versions are (meaning how many transformations have been applied). If an explanation is not found for a specific change, it is declared idiopathic (unexplained).

Note that among the different changes, the vertical additions are the most common and complex and, thus, the presented methods mostly address such a scenario. Specifically, for the presentation of methods, we assume the goal as a single attribute (a right-hand attribute to be explained) with its data values. Also, given a goal, finding its origin is not straight forward. For the moment, assume the origin is the original left-hand table T. We discuss a method to "find" an origin, given a goal, in Section 4.6.

4.1 Numeric Change Explanations

Whenever we need to explain a numeric goal using a relation origin that hold numeric data, we position the problem as *regression* in which the origin relation tuples are treated as independent variables and the goal relation tuples as dependent variables. Aiming at explainable transformations, we build on top of linear regression [29]. To reduce model complexity and prevent over-fitting [93], we experiment with Lasso and Ridge regularization.

Example 6. *A numeric transformation is given in Figure 1b, where explaining* a6 *can be resolved by fitting a regressor* $\frac{1}{60} \cdot a2$.

Not all numeric transformations can be covered by a linear function. Accordingly, to allow richer, more flexible, numeric transformations, we *extend* the feature space (i.e., the origin) by generating additional features. Note that while these extensions are motivated by commonly used data science and engineering operations [64], they do not (and cannot) cover every possible transformation.

Polynomial Regression and Inter-relation Features: To explain polynomial transformations, we generate additional polynomial features [45] over $\mathcal{O}$. Given a predefined degree d, the polynomial extension of $\mathcal{O}$ is given by $poly(\mathcal{O})$ whose attributes correspond to $\{A_i^2, \ldots, A_i^d, \forall A_i \in \mathcal{O}_{name}\}$. The extended origin relation is created on a tuple level by applying the associated operation. For example, the attribute A_i^2 of the tuple r_j in the extended relation would get the value in attribute A_i squared, meaning $\pi_{A_i}[r_j]^2$. We also introduce feature inter-relation, that is, multiplication and division between different attribute values in $\mathcal{O}$. Note that addition and subtraction are already supported when using linear regression. The inter-relation extension of $\mathcal{O}$, $inter(\mathcal{O})$ corresponds to $\{A_i \cdot A_j, A_i \div A_j \ldots, \forall A_i, A_j \in \mathcal{O} \wedge A_i \neq A_j\}$. Also here the transformations are done on the tuple level, e.g., the attribute $A_i \cdot A_j$ of the tuple r_j would get the value $\pi_{A_i}[r_j] \cdot \pi_{A_j}[r_j]$. The extensions can also be applied consecutively, e.g., $inter(poly(\mathcal{O}))$ to create attributes such as $A_i \div A_j^2$ to resolve, for example, the BMI formula ($kg \div m^2$).

Mathematical Transformations: When generating new features over numeric data, it is also common to use mathematical operations [26]. Specifically, to support this type of explanation we generate a math extension of $\mathcal{O}$, $math(\mathcal{O})$, with the attributes $\{log(A_i), sqrt(A_i), reciprocal(A_i), exp(A_i), \ldots, \forall A_i \in \mathcal{O}\}$, where $sqrt(A_i) = \sqrt{A_i}$, $reciprocal(A_i) = A_i^{-1}$, and $exp(A_i) = e^{A_i}$. The transformations are tuple-based, e.g., the feature $log(A_i)$ of the tuple r_j in the extended relation would get the value $log(\pi_{A_i}[r_j])$.

Global Aggregations: We also generate aggregate features. This extended set is especially important when looking at one of the most common transformations in machine learning, that is, (value) normalization [102]. As above, we introduce an extension of $\mathcal{O}$, $agg(\mathcal{O})$, with the following attributes $\{sum(A_i), mean(A_i), max(A_i), min(A_i) \ldots, \forall A_i \in \mathcal{O}\}$. Here, although assigned on a tuple level, the extended values are computed over all the values in the attribute. For example, the feature $sum(A_i)$ of the tuple r_j would get the value $\sum_{r_{ik} \in \pi_{A_i(T)}} r_{ik}$. Then, if a user applies a sum normalization over A_i, the feature-set $agg(inter(\mathcal{O}))$ that includes the feature $A_i \div sum(A_i)$ would be able to resolve and explain this added attribute. Similarly, in the case of min-max normalization [102] the additional features of $min(A_i)$ and $max(A_i)$ can be used to generate accurate explanations. Another example that can be solved using $agg(\mathcal{O})$ is the common collaborative filtering transformation of subtracting the mean value ($A_i - mean(A_i)$) [68].

An extended feature-set is used to fit a regressor that assigns coefficients for extended feature-set. A perfect regressor would be able to deal one all-inclusive feature-set (i.e., $poly(inter(\ldots(\mathcal{O}) \cup inter(poly(\ldots(\mathcal{O}), \ldots)$; yet, since it is not realistic to expect that (from an explainable regressor), we apply each set independently (power-set of the extensions) and generate multiple "possible" explanations. Section 7.4 addresses the issue of choosing among them.

Note that the transformation is "learned" (fitted) only based on the two dataset versions T and T' and **no additional training data** is required. Section 5.1 discusses the main application of numeric change explanations and illustrates its using multiple examples.

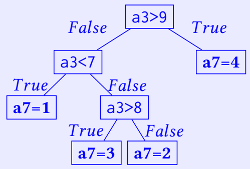
4.2 Categorical Change Explanations

When we aim to explain a categorical goal using an origin relation that hold numeric data, we position the problem as *classification*, in which the tuples of the origin relation are treated as explanatory variables and the goal relation tuples are used as the output class labels. The output class can be binary (e.g., is movie longer than two hours) or multi-class (e.g., a7, Example 1). Aiming at explainability, we focus on decision trees [29].

[3] Data-driven reflects that we only use data values (we do not use meta-data).

[4] Explain-Da-V can be easily extended to support additional types such as dates.

Example 7. *Figure 1b provides an example a categorical transformation, namely,* a7*, which can be resolved with the help of the following decision tree.*



Note that decision trees cover only explanations that can be represented as disjunctions of conjunctions [71], where each path from the root to a leaf corresponds to a conjunction and the tree itself is the disjunction of these conjunctions.

4.3 Textual Change Explanations

When the origin and/or goal are textual, we follow the PBE approach (see Section 2.3), using a search-based solution. Specifically, we adopt an existing framework called Foofah [57]. The PBE solution is composed of designing a space of possible operators and a search algorithm. The search algorithm (A*, following Foofah) navigates the space of operators using a heuristic function (based on dissimilarity of tables) that estimates the cost of any proposed partial solution. The space is pruned to boost search speed [57].

4.3.1 Textual-to-Textual. Addressing data versioning, we extend the traditional PBE operators to include operators the cover frequent text-processing steps [96], including text lowering, lemmatization, removal of special charterers (e.g., punctuations and numeric values) and tokens (e.g., stop-words and html tags). All implemented additional operators for foofah are given in our repository [6].

Unlike Foofah, we also consider textual-to-numeric and textual-to-categorical. Note that although in recent years transformer-based models have become a standard way to extract (latent) features from text, traditional feature engineering over text, that is, extracting manual numeric and categorical features from textual values, is still an important ingredient in NLP [36, 39, 63] and in other research disciplines such as HCI [30] and information management [50].

4.3.2 Textual-to-Numeric. The search space defined for resolving this kind of transformation includes a meta-operation that counts the occurrences of some pattern *pat* in a value ($\texttt{count}_{pat}(r_{ij})$), where r_{ij} is a value in the table, see Section 3). Using this operation we can define operations such as number_of_words = $\texttt{count}_{`\ '}(r_{ij})$ and number_of_questions = $\texttt{count}_{`?'}(r_{ij})$. We also cover counting a pre-defined set of stop-words and punctuation marks.

4.3.3 Textual-to-Categorical. We define a similar meta-operation for a pattern existence ($\texttt{contains}_{pat}(r_{ij})$), which is used to generate operations such as contains_percent = $\texttt{contains}_{`\%'}(r_{ij})$.

Example 8. *Figure 1b provides an example of textual-to-textual transformation, namely,* a5*, which can be resolved with the help of the foofah environment in its original implementation. Specifically, if we consider* a1 *as an origin, foofah would consider* $\pi_{a1}[T]$ *as input examples and* $\pi_{a5}[T']$ *as corresponding output examples (goal in our terms) and synthesize a tuple-based data transformation program* (1) t = split(t, 0, '('), (2) t = split(t, 1, ')'), (3) t = drop(t, 0), (4) t = drop(t, 2) *the tuple value* Moana (U)*, for example, would be transformed as follows* [Moana , U)] →[Moana , U,] → [U,] → U.

Resolving a8 *requires* Explain-Da-V*'s extensions that includes textual-to-numeric transformations (*len()*).*

Note that Foofah aims to synthesize transformations using a given set of example tuples, and is often able to do so using just a few examples. Our goal, in contrast, is to generate an explanation that correctly explains a full table transformation (a dataset version). Hence, our tremendous expansion of the search space beyond text-to-text transformations plays a critical role. Moreover, Section 4.6 introduces a technique that prunes the search space in this context.

4.4 Categorical-encoding Change Explanations

Whenever dealing mixed types, that also contain some textual values, these values may be encoded. A common encoding approach, which we use here, is one-hot-encoding [83]. Let $A_i \in O$ be a textual attribute in the origin. One-hot-encoding of this attribute generates an additional attribute for each unique value (or category) in A_i and assigns a value of 1 to each tuple that corresponds to this value (category). This addition, not only allows the resolution of this common encoding scheme, but also a richer representation that can be used to resolve other types of encoding (e.g., ordinal encoding) and additional transformations. Such encodings are also commonly used in data preparation for machine learning [27]. Recalling Example 1, if a user aims to predict the rating of a movie, extracting features such as Is_Drama or Is_Action can be beneficial for learning.

4.5 Reshaping Change Explanations

Generally group-by is a table-reshaping operation [100], i.e., a natural attribute-match and tuple-match do not exist. However, when it comes to feature engineering, group-by can also be used to generate aggregated features based on some other attribute. The latter is addressed in a manner that is similar to numeric change explanations. An extended origin, similar to Section 4.1, would be created for each numeric attribute $A_j \in T_A$ with respect to each textual/categorical attribute $A_i \in O$ independently or conjointly (grouping by multiple attributes). We use an SQL syntax for clarity. A helper query (Figure 3) can be used to generate the extended group-by features of the numeric attribute $A_j \in T_A$ with respect to a textual/categorical attribute $A_i \in O$.

```
SELECT A_i, mean(A_j), max(A_j)...
FROM T
GROUP BY A_i
```

Figure 3: Group by Query

If more than one numeric attribute exists, it will be added to the GROUP BY and SELECT clauses. Using this helper query, by joining it with the origin, we obtain the additional possible attributes.

We also consider the reshaping scenario introduced in previous work, e.g., [100]. Reshaping is often considered as a possible operation that can be applied over a table throughout the search (see Section 2.3). We introduce an alternative data-driven approach. Specifically, reshaping is associated only with attribute-match when there is no tuple-match. We explicitly reshape the table using the query of Figure 3 and fit a regressor over it. Currently, Explain-Da-V does not support other reshaping transformations such as transpose and pivot, which we intend to explore in future work.

4.6 Finding the Origin

A naïve solution to use all available data values. For example, in the context of adding attributes, using all attributes of T as an origin, i.e., $O = (T_A, T)$. The problem with this method is twofold. First, unrelated attributes may serve as noise when aiming to find a proper transformation for the goal. For example, referring back to Example 1 , aiming to resolve a5 using all attributes (a1-a4) presents much more noise than aiming to resolve it using a2. A second issue has to do with the data types. Utilizing a numeric change explanation (Section 4.1) may be more beneficial than a textual change explanation (Section 4.3). For example, using a2 as an origin to explain a5 instead of using a1 to a4.

When examining the creation of a new attribute from existing attributes, we observe a side effect of creating a *functional dependency* between the origin and the new attribute (goal). For example, if two movies have the same runtime in minutes (a2), they will have the same runtime in hours (a5), which, by definition constructs a functional dependency between a2 and a5. Accordingly, we use a functional dependency discovery algorithm [79] to find the origin.[5] The dependencies we are interested in are the ones in which our goal is the dependent set and the discovered determinant is used as the origin. Note that there can be more than one attribute set that determines the goal and accordingly multiple origins are generated.

We analyze all determining attribute sets by considering each one of them as a candidate origin. Accordingly, multiple explanations may be generated for a goal. Section 7.4 describes how we choose among them. Specifically, we rank the determinants by size and cardinality and, if desired, an early stop condition can be introduced based on the size or quality of the discovered transformation.

Example 9. *Recall Figure 1 and consider attribute* a7 *as a goal. Since the example tables are small, any combination of attributes in {*a1, a2, a3, a4*} can be considered as an origin. If no high quality explanations are found for singleton attributes, the algorithm can consider combinations of attributes. In a larger real example, only a few attributes or combinations of them may be an origin.*

5 EXPLAINING VERTICAL CHANGES

With our arsenal of explanation methods, we now consider how to use them to explain changes. We begin with attribute additions, after which, we describe our approach to handling attribute removal.

5.1 Addition Explanations for $R\Delta_A$

Adding an attribute is a very common operation in data science, mainly revolving around data preparation and feature engineering for machine learning (ML) [102]. Added attributes are usually a result of applying some transformation over the existing data. We first find the origin (Section 4.6) an then utilize the core explanation methods (Section 4) to resolve the transformation that, when applied to the origin relation (input), generates the desired goal relation (output). Figure 4 provides a sketch of the approach (left and middle parts) and highlights its main novelties (right part) in the context of adding attributes.

Explain-Da-V attribute addition explanation is iterative, aiming to resolve each added attribute (i.e., the goal $\mathcal{G} = (A_i, \pi_{A_i}[T'])$) at a time. First a set of possible origins for A_i is found following Section 4.6. Then, Explain-Da-V utilizes the core explanation methods (Section 4) in a case-based manner (according to the types of the origin and the goal) to generate explanations. For example, if a numeric origin is found for a numeric goal, Explain-Da-V uses a numeric explanation method (Section 4.1). Multiple explanations are generated for each goal (e.g., due to multiple origins), for which we introduce a search strategy in Section 7.4. A full algorithm is given in a technical report [88]. We illustrate it over Figure 5, which provides an additional version of the table in Figure 1a. A more detailed example can be found in a technical report [88].

[5] In our experiments we use a discovery algorithm called FDEP [48].

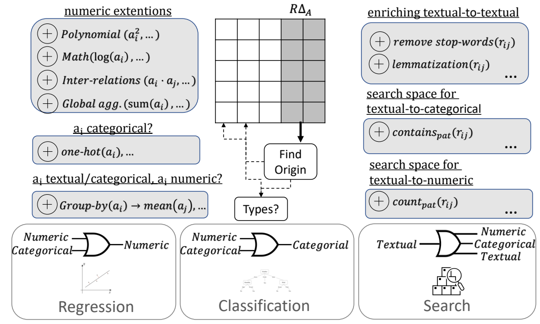


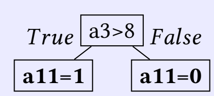
Figure 4: Explain-Da-V explanation of attribute addition.

a0	a1	a2	a3	a4	a9	a10	a11	a12	a13	a14
m1	The Godfather (A)	175	9.2	Drama	0.28	3.15	1	godfather a	8.9	1
m2	Hamilton (PG-13)	160	8.6	Drama	0.28	3.23	1	hamilton pg	8.9	1
m3	The Avengers (UA)	143	8.0	Action	0.24	3.36	0	avengers ua	8.0	2
m5	Moana (U)	107	7.6	Animation	0.23	4.26	0	moana u	7.6	3

Figure 5: Dataset version created by UserC over Figure 1a.

Example 10. *Among the new added attributes* a9 *and* a10 *are numeric. The attribute* a9 *is a (sum) normalization of the values in* a3 *(normalized rating).* Explain-Da-V *would first find its origins (Section 4.6). As in the case of* a6 *(see Example 9), also* a9 *can be determined by multiple attribute sets. Among the possible origins, consider* a3*. Since both* a3 *and* a9 *are numeric,* Explain-Da-V *uses numeric change explanations (Section 4.1). A baseline explanation will be generated by fitting a regressor over* a3*. Then, different extensions will be applied over the origin, each will be associated with an explanation by fitting a regressor. Among the generated extensions we will find* $agg(inter(O))$*, that contains the feature* a3÷sum(a3) *over which a transformation* a9=a3÷sum(a3) *will be fitted to generate an explanation* $\mathcal{E}_{a9} = (a3, a3{\div}sum(a3))$*. Similarly, consider {*a2, a3*} as an origin for* a10*, the following explanation will be generated* $\mathcal{E}_{a10} = (\{a2, a3\}, 60 \cdot a3 \div a2)$ *(rating per hour).*

Example 10 (cont.). *Attribute* a11 *is categorical and consider, for example, the numeric origin* a3*. Accordingly,* Explain-Da-V *uses categorical change explanations (Section 4.2) and might fit the following decision tree to explain* a11*. Note that* a11 *might been created by applying* a3>8.5 *which differs from* Explain-Da-V*'s data-driven explanation. Our goal however is to provide an accurate explanation which both are. Interestingly, if we consider* a2 *as an origin we can derive a similar decision tree rooted at* $a2 > 150$*. These issues refer to the explanations generalizability, which is discussed in Section 7.*



Attribute a12 *is textual and focuses on text cleaning.* Explain-Da-V *executes Section 4.3.1 and using, for example,* a1 *as an origin, the resolved transformation removes punctuation marks (e.g., '(') and numeric values ('13') and lowers the text. Note that such a transformation requires our extensions to Foofah and would not be accurately resolved using the original Foofah [57].*

Attribute a13 *is numeric and among possible origins, consider {*a3*,*a4*}, which is mixed.* Explain-Da-V *would turn to encoding (Section 4.4) and reshaping (Line 4.5). Consider the latter,* Explain-Da-V *will generate an explanation using the transformation* $1 \cdot (mean(a3)\ by\ a4)$ *which represents a group by* a4 *and computing the mean of* a3 *(mean rating by genre).*

Finally, attribute a14 *is categorical and consider,* a4 *as an origin. In addition to applying trying to find textual explanations (Section 4.3.3),* Explain-Da-V *would also turn to encoding (Section 4.4) and reshaping (Section 4.5) explanations. Consider the former and note that* a14 *is an ordinal encoding of attribute* a4 *(Drama⟶1, Action⟶2, Animation⟶3). The three encoded attributes, namely* is_Drama*,* is_Action? *and* is_Animation? *are used to resolve* a14*. Note that in a real-world scenario, an attribute like* a14 *would not necessarily be recognized as a categorical (e.g., high cardinality or misclassification as a numeric value). In this case* Explain-Da-V *would turn to Section 4.1 resulting in the transformation* 1·is_Drama? + 2·is_Action? + 3·is_Animation?*.*

5.2 Removal Explanations for $L\Delta_A$

Removing attributes is less common and usually include superficial transformations. We treat each attribute in $L\Delta_A$separately as a goal. We cover two main types of explanations for removal reflecting data cleaning (removing duplicated and noisy attributes).

First, we examine a *table-independent* attribute removal, which in our terms reflects an empty origin ($\mathcal{O} = \emptyset$). Specifically, we use a threshold to decide whether a attribute was removed because it has too many (above a threshold) missing (NaN) values.[6] In this case an explanation for a removed attribute $A_i \in L\Delta_A$ will be in the form of $\mathcal{E}_{A_i} = (\emptyset, \text{'contains missing information'})$. Formally, the 'contains missing information' can be defined as $A_i = \begin{cases} \emptyset, & \text{if ratio of NaN values} > \alpha \\ A_i, & \text{otherwise} \end{cases}$,
where $\alpha \in [0, 1]$ is some threshold.

As a second case, we look into duplicated information. A trivial explanation can be provided for an identical attribute in T'. Given a goal A_i, the origin is some attribute $A'_j \in T'$ such that $A'_j \notin T$ and $\pi_{A_i}[T] = \pi_{A'_j}[T']$ (full overlap of values). A natural extension of finding duplications is looking into similarities between attributes. We look into two types of similarities, measuring the overlap between attributes and if there is a one-to-one dependency between them. Overlap is measured and, if it meets some threshold, an explanation is generated using the overlapping attribute A'_j as the origin and an 'overlaps with A'_j' transformation, which is defined similar to above. We also check if some attribute in T' determines (using a similar methodology as described in Section 4.6) $A_i \in L\Delta_A$. Obviously many other measures of similarity exist, which we intend to explore in future work. Finally, note that sometimes attribute removal can be idiopathic, i.e., the user simply removed an attribute because they are not interested in some parts of the data.

[6]The threshold can be treated as an hyper-parameter or a user-provided input.

6 EXPLAINING HORIZONTAL CHANGES

We begin with the common data cleaning operation of tuple removal, after which we discuss adding tuples.

6.1 Removal Explanations for $L\Delta_r$

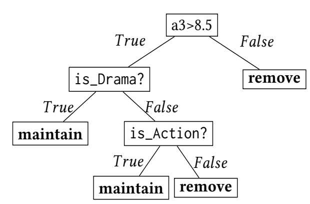
Tuple removal is a very common operation in data preparation, which mainly revolves around cleaning data. Our examination begins iteratively by looking into each removed tuple in $L\Delta_r$ independently. This may, for example, result in the horizontal explanation $\mathcal{E}_{m4}$ from Example 5. Finally, we explore if a *predicate* was applied to remove them all remaining (unexplained) tuples conjointly.

As in Section 5.2, we aim to find tuples that were removed collectively in a table-independent manner due to missing values (NaNs), see the m4 explanation in Example 5. For table-dependent explanations, we aim to find duplicated tuples, which is a common result of data cleaning using entity resolution, which is a rich and active research [47, 67]. We focus only on identical tuple removal. Note that this strict requirement can be relaxed and any entity resolution technique, e.g., using declarative rules [28], can be used to find duplicated tuples removal explanations. Specifically, if a duplicated tuple $r'_j \in T'$ is found for a goal tuple $r_i \in L\Delta_r$, we create a horizontal explanation of the form $\mathcal{E}_{r_i} = (\emptyset, \text{duplicated of } r'_j)$. This transformation can be expressed as follows $r_i = \begin{cases} \emptyset, & \exists r'_j \in T' s.t. r_i = r'_j \\ r_i, & \text{otherwise} \end{cases}$
Finally, outlier detectors (Z-method and IQR-method [31, 94]) also serve as explanations for removed tuples.

Not all tuples can be explained independently, thus, for all unexplained tuples, $L\Delta_{r_{unexplained}}$, we aim to find a joint explanation in the form of a predicate. Given a set of unexplained tuples, we use a categorical explanation method (Section 4.2) to find a joint explanation. Similar to Section 5.1, in case the origin has mixed types, the decision tree is applied also over encoded (using categorical-encoding change explanation, Section 4.4) attributes in the table.

a0	a1	a2	a3	a4
m1	The Godfather (A)	175	9.2	Drama
m2	Hamilton (PG-13)	160	8.6	Biography
m3	The Avengers (UA)	143	8.0	Action
m4	Inception (UA)	NaN	8.8	Action
m5	Moana (U)	107	7.6	Animation

(a) Dataset version created by UserD over Figure 1a.



(b) Tree

Figure 6: UserD version of Figure 1a and its explanation.

Example 11. *In the example of Figure 1b we present a simple example of tuple removal due to NaN value. Figure 6a provides an example of applying a predicate over the table. To resolve this predicate,* Explain-Da-V *will first add the one-hot-encoded features corresponding to* a4 *(*is_Drama?*,* is_Action? *and* is_Animation?*), then, using a decision tree, it will try to resolve the predicate. The decision tree in Figure 6b will be generated.*

As mentioned in Section 4.2, only predicates that can be represented as disjunctions of conjunctions can be resolved.

6.2 Addition Explanations for $R\Delta_r$

The non-idiopathic addition of tuples may be a result of oversampling (bootstrapping). To detect such a transformation, we use a similar methodology as in Section 6.1. Given an added tuple $r'_i \in R\Delta_r$, we aim to find a duplicated (equal or similar) tuple $r_j \in T$ to create an explanation noting that the tuple has been bootstrapped.

7 EVALUATING EXPLANATIONS

We aim to generate user friendly explanations that capture the semantics of changes. Specifically, the explanation (transformation) can reproduce the change and generalize it beyond a specific setup. Aiming to assess such semantics, we now describe how we evaluate explanations. Sometimes multiple explanations can be generated with respect to a change. Recall Example 10 in which we present two possible valid explanations for a11. The first decision tree explanation is rooted at $a3 > 8$ and the second is rooted at $a2 > 150$. Also a decision tree rooted at $a3 > 7.5$ is a possible (invalid) explanation. In what follows, an important question that needs to be asked is *how to compare (and choose among) possible explanations?*

Related work on data transformation (see Section 2.3) employ success rates that measure whether the output was generated successfully by applying the transformation over the input. Yang et al. also introduce a ranking measure (MRR) over possible transformations (pipelines in their terms), which still views the transformation as a whole [100]. We claim that solely using such a measure does not capture the true nature of the transformation, especially when evaluating an attribute-to-attribute (Section 5.1) transformations. To provide a more fine-grained evaluation, we evaluate both the *validity* and *generalizablity* computed over the transformed values to assess the coverage of the transformation. As we are interested in providing explainable solutions, we also use two *explainability* dimensions, *conciseness* and *concentration*.

7.1 Explanation Validity and Generalizablity

Attribute (vertical) addition explanations are richer than attribute removal or tuple transformations, so their evaluation is addressed accordingly.

Vertical Additions: We separate this evaluation into validity (*does the generated transformation recreate the goal using the origin?*) and generalizability (*will the generated transformation be able to recreate a similar goal using a similar origin?*). Recall Definition 3 and the notation of origin ($\mathcal{O}$), goal ($\mathcal{G}$), and transformation ($\mathcal{P}$). For simplicity, we denote the output of a transformation applied to an origin relation as $\hat{\mathcal{G}} = \mathcal{P}(\mathcal{O}_{relation})$. Validity is computed in a tuple-based manner over value-pairs $(\hat{r}_{ij}, r_{ij})$ such that $\hat{r}_{ij} \in \hat{\mathcal{G}}$ is a transformed value corresponding to a goal value $r_{ij} \in \mathcal{G}_{relation}$, i.e., $\hat{r}_{i0} = r_{i0}$ (r_{i0} is the tuple id so this means the tuples are matching, see Section 3). Explanation validity is measured as follows:

$$Val(\mathcal{E}_{\mathcal{G}}) = \frac{1}{|\mathcal{G}_{relation}|} \sum_{\substack{\hat{r}_{ij} \in \hat{\mathcal{G}}, r_{ij} \in \mathcal{G}_{relation}: \\ s.t. \hat{r}_{i0} = r_{i0}}} \mathbb{I}(\hat{r}_{ij} = r_{ij}) \tag{1}$$

where $\mathbb{I}(\hat{r}_{ij} = r_{ij})$ is an indicator returning the value 1 if the transformed value equals to the corresponding goal value and 0 otherwise. The validity can be viewed as a tuple-based success rate, i.e., the proportion of the tuples that were successfully transformed.

Example 12. *Recall the vertical explanation $\mathcal{E}_{a9} = (a3, a3 \div sum(a3))$ which was created for the attribute* a9 *in Figure 5 (see Example 6). Also consider an alternative vertical explanation $\mathcal{E}'_{a9} = (a3, a3 \div 33.4)$. Both explanations would have a validity score of* 1 *as applying the corresponding transformation recreates* a9 *perfectly.*

As illustrated in the example, validity only looks at the given dataset versions T and T', which may result in overfitting (e.g., selecting $a3 \div 33.4$ over $a3 \div sum(a3)$). Aiming to measure such scenarios, we now introduce generalizability, measuring the extent to which a generated solution can explain an equivalent set of versions. Specifically, generalizability can be measured if a pair of versions $\tilde{T}$ and $\tilde{T}'$ exist such that $\tilde{T}'$ was generated as a version of $\tilde{T}$ using the same transformations that were used to generate the version T' from T. Let $\tilde{\mathcal{O}}$ be the origin over $\tilde{T}$ and $\tilde{\mathcal{G}}$ the goal over $\tilde{T}'$. The generalizability of an explanation $Gen(\mathcal{E}_{\mathcal{G}})$ is measured by applying $\mathcal{P}$ over $\tilde{\mathcal{O}}$ to generate $\hat{\tilde{\mathcal{G}}}$ and is computed as in Eq. 1 over $\hat{\tilde{\mathcal{G}}}$ and $\tilde{\mathcal{G}}$. When designing our newbenchmark (Section 8.1.1), we generate an annotated *hold-out* set, which is not available for Explain-Da-V and used it to compute generalizability. We illustrate the importance of generalizability using the following example.

a0	a1	a2	a3	a4
m6	Pulp Fiction (A)	154	8.9	Drama
m7	Saw (UA)	103	7.6	Horror
m8	Snatch (UA)	104	NaN	Crime
m9	King Kong (Passed)	100	7.9	Adventure

a0	a1	a2	a3	a4	a9	a10	a11	a12	a13	a14
m6	Pulp Fiction (A)	154	8.9	Drama	0.36	3.47	1	pulp fiction a	8.9	1
m7	Saw (UA)	103	7.6	Horror	0.31	4.43	0	saw ua	7.6	4
m9	King Kong (Passed)	100	7.9	Adventure	0.32	4.74	0	king kong passed	7.9	5

Figure 7: Example versions for generalizability

Example 13. *Fig. 7 provides two dataset versions. The top table is similar to Fig. 1a and the bottom table corresponds to Fig. 5 such that the same transformations over Fig. 1a generates Fig. 5. Recall the explanations $\mathcal{E}_{a9} = (a3, a3 \div sum(a3))$ and $\mathcal{E}'_{a9} = (a3, a3 \div 33.4)$. While these two are valid, using Fig. 7, we observe that $\mathcal{E}_{a9}$ is also generalizable while $\mathcal{E}'_{a9}$ is not. Specifically, if we apply $\mathcal{E}_{a9}$ over* a3 *in Fig. 7 we obtain the values of* a9 *in the bottom table. However, if we apply $\mathcal{E}'_{a9}$, we obtain the values* 0.27, 0.23, *and* 0.24 *for the records* m6, m7, *and* m9, *respectively, resulting in a* 0 *generalizability.*

Other Explanations: For tuple removal, we apply a reconstruction methodology to evaluate validation and generalizability globally. We gather all generated explanations and apply them over T and try to regenerate T'. Then, we check the overlap between the removed tuples and the tuples that are not included in T'. For example, if a a3>8 predicate was used to explain the removed tuples, the same predicate would be applied over T and compared to T'. This overlap, i.e., the proportion of tuples that were correctly removed using the explanations of Explain-Da-V, is used as the overall validity of tuple removal. The generalizability is measured similar to above using an additional dataset version pair $\tilde{T}$ and $\tilde{T}'$.

We also compute validity and generalizability for other explanations. Validation and generalizability of a removed attribute or added tuple are computed independently, i.e., a score of 1 is given if an attribute was removed correctly or a tuple was added correctly.

7.2 Problem Definition

Using the aforementioned measures, we formally state the problem of explaining data versions. Recall the change sets defined over the dataset versions T and T' (see Section 3.2).

Definition 14. *Given T', a version of T where the goals are $L\Delta_A$, $R\Delta_A$, $L\Delta_r$, and $R\Delta_r$ meaning the left-hand delta attributes, right-hand delta attributes, left-hand delta tuples, and right-hand delta tuples between them. From a search space of possible explanations, the version explanation problem is to find for each goal a set of explanations with the highest validity.*

Hence, a solution to the version explanation problem is a set of explanations that composed come closest to producing T' from T. Note that with validity alone we may have ties (as in our examples where multiple explanations have validity 1). Among multiple solutions to the problem, we also aim to choose "explanable" explanation, that is, explanations that a user can better understand.

7.3 Explanation Explainability

As motivated above, we care about the explainability of the generated solution. We, again, mainly focus on the attribute addition transformations. Since we use different models to generate explanations (regressors, decision trees, and programs), we seek a common ground to measure explainability. Inspired by Narayanan et al. [73] and Lakkaraju et al. [65], who focus on decision sets, we introduce two explainability dimensions, namely *conciseness*, and *concentration*, that can be measured across different explanations types.

Explainability Conciseness: Studies show that the fewer the components in a model and the shorter it is, the easier it is for a user to understand it [38, 80]. In what follows, we measure the conciseness of the transformation as the number of components (N_c) it holds. For regression models we use the *number of coefficients*, for decision trees we use the *number of nodes*, and for programs we use the *number of implementation lines*.

Example 15. *As a simple example, consider an $exp(A_i)$ transformation. Obviously a desired explanation would use the $math(\cdot)$ extension (using Section 4.1) to generate an explanation $(A_i, exp(A_i))$ that, in addition to being valid and generalizable, obtains an explainability conciseness of 1 (a sole coefficient with the value 1).*

An alternative explanation would use a Taylor Series over the $poly(\cdot)$ extension to generate a valid and generalizable explanation $(A_i, 1 + A_i + \frac{A_i^2}{2} + \frac{A_i^3}{6} + \dots)$ with an explanability conciseness of $\frac{1}{d+1}$, where d is the polynomial degree. Note that this case also highlights the trade-off between validity (or generalizability) and explainability. The bigger the selected degree, the higher the validity (and generalizability) and the lower the explainability.

Explainability Concentration: While a more concise explanation is favorable, it should also contain as few components as possible [66] (i.e., it should be as concentrated as possible). Specifically, since humans have a limited working memory, a solution that is grouped into fewer chunks of information is favorable [73]. For example, a linear regression function is easier to understand than a polynomial regression with reciprocal and logarithmic transformations, even if the former is longer. For regressors, we count *the extensions that were used* (e.g., polynomials and math operations). For decision trees, we count *the number of internal nodes* that represent conditions and for programs we use *the number of intermediate transformations*. Let N_g be the number of chunks, the explanability concentration is then given as $1 \div N_g$ such that a more concentrated transformation gets a higher score.

Example 16. *Recall Example 15 and the explanations $(A_i, exp(A_i))$ and $(A_i, 1 + A_i + \frac{A_i^2}{2} + \dots)$. The concentration of these explanations is 0.5 (1 extension, 1 degree) and $\frac{1}{d}$, respectively.*

To highlight the difference between conciseness and concentration consider, for example, $\mathcal{E}_1 = A_1 + A_2 + 5$ and $\mathcal{E}_2 = log(A_1) + A_2^2 \cdot A_1$. While $\mathcal{E}_1$ is less concise ($\frac{1}{3}$ vs $\frac{1}{2}$ of $\mathcal{E}_2$), it is more concentrated (1) than $\mathcal{E}_2$ ($\frac{1}{3}$) which involves two additional extensions.

The *total explanability* is a linear combination of conciseness and concentration that can be defined by a user or a system.[7]

7.4 On Choosing an Explanation

`Explain-Da-V` works iteratively, aiming to find valid explanations for each detected change following Section 5 and Section 6. As mentioned above, for each goal, multiple explanations can be generated, for example, if there are multiple origins (Section 4.6) or we have more than one methodology to explain a transformation (e.g., different extensions in Section 4.1). `Explain-Da-V` chooses the most *explainable valid* explanation for each goal.

Each independent explanation is derived in a way that optimizes some notion of error within the respective context that is not always the same as our definition of validity. A regressor (Section 4.1) minimizes the mean squared errors, a PBE solution (Section 4.3) directly optimizes accuracy via search and a decision tree (greedily) optimizes the split functions of nodes. Given a set of explanations, we choose one as follows. (1) Find the highest validity in the set. (2) If multiple explanations share this value, return the most explainable based on total explainability (see Section 7.3).

Note that generalizability can not be used for explanation selection unless we have access to a $\tilde{T}$ and $\tilde{T}'$ (see Section 7.1).

Potentially, there can be a large number of transformations. Dealing with this size, the explanations in a set are generated in a sorted order by the size and cardinally of their origin (see Section 4.6). Similarly, among regression models the explanations are sorted by the amount of extensions that were applied (i.e., first, a model without extensions is considered). Accordingly, we introduce an *early stop condition* such that if an explanation meets a predefined threshold of validity and explainability, it is returned and the search is stopped.[8] Empirically, almost 70% of cases are terminated early.

8 EMPIRICAL EVALUATION

We now compare its performance to baselines (Section 8.2) and analyze its components using an ablation study (Section 8.3).

[7] In our experiments we use a uniform combination.

[8] In our experiments the threshold was set to .95.

Table 2: Semantic Data Versioning Benchmark Details.

Topic (Name)	# of Original Tuples	# of Original Attributes	# of Versions	# of Version-pairs
Movies and TV shows [8] (IMDB)	1,000	6	72	29
NBA Players [10] (NBA)	11,700	9	68	27
Wines Reviews [12] (WINE)	129,971	6	72	29
Iris Flowers [9] (IRIS)	150	5	58	22
Titanic Passengers [11] (TITANIC)	891	6	72	29

8.1 Experimental Setup

We now detail our benchmarks, implementation, and baselines.

8.1.1 Benchmarks. We design a new benchmark for the novel task of semantic data versioning, termed Semantic Data Versioning Benchmark (SDVB), composed of five *version-sets*. We also adopt a publicly available dataset designed by Yang et al. [100] for a similar task of synthesizing data pipelines.

Semantic Data Versioning Benchmark (SDVB): SDVB contains a total of 342 dataset versions (136 version pairs) over five different topics, ranging in length (number of tuples) and width (number of attributes).[9] Each topic represents a *version-set* that was derived from a well-known seed dataset detailed in Table 2, which includes smaller datasets (e.g., IRIS) along side bigger datasets (e.g., WINE). **Version Generation:** Given a seed dataset, we revise it to generate a version of it by first selecting a subset of change dimensions (e.g., $R\Delta_A$and $L\Delta_r$). Then, based on the dimension, we perform a set of transformations (some sampled and some manually created). We provide a benchmark sample and its generation notebook in our repository [15].[10] We assure that each of the five version-sets cover all change dimensions. Prior to version generating, each dataset is split into T and $\tilde{T}$ (80%-20%), where the latter is a hold-out to compute generalizability. Following Section 7.1, the same changes applied to T to generate T' are applied to $\tilde{T}$ to generate $\tilde{T}'$.[11]

Finally, note that a version may be created using more than one change and, in practice, the aforementioned number of versions is actually **composed of 1,702 changes**. For example, to create the 72 WINE dataset versions, a total of 681 changes were applied over the original dataset and its versions.

Auto-Pipeline Benchmark [1]: This benchmark contains real data pipelines extracted from Github notebooks. As we focus on dataset versions, we filter out pipelines that include more than one table (e.g., those that use a join). Following Yang et al. [100], we consider the "test" table as T and the "target" table as T'. For a fair comparison, we run `Explain-Da-V` and all baselines on all the data and do not consider generalizability for this benchmark.

8.1.2 Implementation. `Explain-Da-V` was implemented in python, following Sections 5 and 6. Main parts of the code are provided in our repository [15]. Linear regression with Lasso [13] and Rigde [16],[12] regularization and decision trees [3] were implemented with Scikit-learn. We extended Foofah's python publicly available implementation [7]. We use the Featuretools [5] framework to generate aggregated and group by features (see Section 5.1).

[9] Not all versions use all original attributes.
[10] The full SDVB and its generation code will become available upon acceptance.
[11] The numbers reported in Table 2 include the hold-outs.
[12] We first tried applying Lasso and if failed we applied Rigde.

8.1.3 Baselines. **Foofah** [57] is used as a PBE baseline (see Section 2.3). **Foofah+** denotes Foofah with our novel extensions (e.g., textual-to-numeric, see Section 4.3). As Auto-pipeline's implementation is not publicly available, we reproduced its search methodology[13] using Foofah's framework by implementing the operators provided by Yang et al. (**Auto-pipeline***) [100]. Search has an exponential worst case time complexity, so we apply a 60 second timeout for all methods following the default in Foofah [57].[14]

As another PBE baseline, we also ran AutoPandas [18] using their publicly available implementation [2]. AutoPandas creates a search space based on pandas [14] operations and prunes the space of programs using deep learning. Similar to the reported performance in Auto-pipeline [100], AutoPandas performance was inferior and thus not reported. We also experimented with SQUARES [77], a recent query reverse engineering framework, and, similarly, do not report its inferior results. Since SQUARES was designed to synthesize traditional SQL queries it can sometimes resolve selection predicates; yet, it fails to cope with other change dimensions such as attributes added using transformations.

Finally, a naïve implementation of the baselines would use all of T and T' as input-output examples. However, to allow a fair comparison, we "find the origin" (see Section 4.6) for each of the baselines and vertical explanations are solved iteratively (each attribute at a time). For horizontal addition explanations the tuples of T are used as input and the tuples of T' as output (similarly for horizontal removal with T' as input and T as output).

8.1.4 Evaluation Measures. The explanations provided by our baselines are of a single type (programs, not regressors or decision trees), thus, in Section 8.2, we compare the Validity (Val.) and Generalizability (Gen.) of `Explain-Da-V` to the baselines. Since `Explain-Da-V` can return explanations that do not have a validity/generalizability score of 1.0, we also report the proportion of such explanations out of all output explanations. We further report the average number of explanations (# $\mathcal{E}$) from which the method selects the most explainable valid (see Section 7.4). We also compare and report runtimes. Section 8.3 also uses explainability (conciseness and concentration).

8.2 `Explain-Da-V` Compared to Baselines

The comparison between `Explain-Da-V` and the baselines (Section 8.1.3) over the benchmarks (Section 8.1.1) is reported in Table 3.

As displayed, `Explain-Da-V` performs much better than the baselines. This out-performance is mainly due to its ability to cope with varying data types (numeric and categorical in addition to textual). The adapted Auto-pipeline benchmark is an exception where `Explain-Da-V` only performed slightly better than Auto-pipeline* as it was not designed for data versioning. Also, even if we zoom-in only on textual transformations (provided in a technical report [88]), `Explain-Da-V` still out-performs all baselines. Even when we evaluate only the 100% valid/generalizable explanations returned by `Explain-Da-V` (denoted in parenthesis in Table 3), we observe a significant improvement. Comparing across baselines, we observe that extending Foofah (Foofah+) provides an average boost of 9.5% in terms of validity and generalizability, which shows the benefit of extending the search space. All methods select among

[13] Reinforcement learning requires training data, which we assume unavailable.
[14] We note that Auto-pipeline default timeout limit is an hour

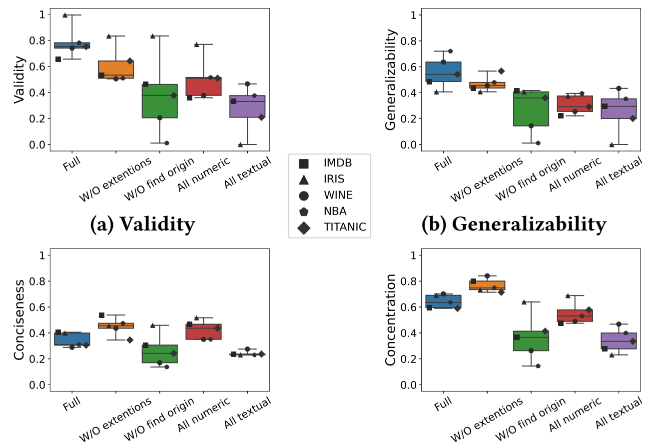
Table 3: Foofah, Foofah+, Auto-pipeline*, and Explain-Da-V performance in terms of Validity (Val), Generalizability (Gen) and average number of explanations the method chooses from (# $\mathcal{E}$). For Explain-Da-V, we also report (in parenthesis) the proportion of explanations with Val/Gen score of 1.

Dataset→	IMDB			NBA			WINE			IRIS			TITANIC			Auto-pipeline		
↓Method	Val	Gen	# $\mathcal{E}$	Val	Gen	# $\mathcal{E}$	Val	Gen	# $\mathcal{E}$	Val	Gen	# $\mathcal{E}$	Val	Gen	# $\mathcal{E}$	Val	Gen	# $\mathcal{E}$
Foofah	.42	.42	3.7	.28	.28	4.2	.29	.29	3.9	.23	.23	3.1	.29	.29	4.1	.55	-	3.3
Foofah+	.44	.44	3.7	.29	.29	4.2	.34	.34	3.9	.25	.25	3.1	.37	.37	4.1	.55	-	3.3
Auto-pipeline*	.44	.44	3.7	.30	.30	4.2	.33	.33	3.9	.26	.26	3.1	.37	.37	4.1	.78	-	3.3
Explain-Da-V	**.73** (.64)	**.60** (.56)	6.4	**.90** (.89)	**.79** (.69)	7.3	**.87** (.76)	**.81** (.59)	6.8	**.93** (.88)	**.83** (.76)	8.9	**.88** (.79)	**.77** (.68)	7.2	**.82** (.78)	-	5.7
+ over baseline	+65%	+36%		+202%	+167%		+156%	+138%		+254%	+217%		+140%	+109%		+5%	-	

multiple explanations (# $\mathcal{E}$, see Section 8.1.4) based on multiple origins (see Section 8.1.3). Explain-Da-V considers almost twice as much explanations since it generates expanded origins for numeric explanations (see Section 4.1).

Comparing among version-sets, we observe that in the IRIS dataset, Explain-Da-V obtained the best performance (.927 Val. and .831 Gen.) and the highest improvement. For the IMDB version-set, Explain-Da-V obtained the worst performance (.732 Val. and .602 Gen.) and lowest improvement (among the newly suggested benchmark version-sets). IRIS is mostly composed of numeric attributes (4 out of 5) which are solved using our numeric change explanations (Section 4.1) and are not dealt with by the baselines. Note that accordingly, Explain-Da-V considers almost three times as many explanations. Yet, the numeric extensions introduced in Section 4.1 and the fact we find the origin helps to home in on a valid solution quite quickly (see runtime below). The IMDB version-set, on the other hand, contains more textual attributes (5 out of 6) and involves changes that Explain-Da-V fails to solve. For example, one of the IMDB version-sets involves a transformation that adds an attribute containing the count of the number of genres from a Genre attribute. In the Genre attribute, the genres are separated by a comma (e.g., Drama, Romance). A correct transformation would, for example, count the number of commas and add 1. While finding a transformation that counts the number of commas is a practical task for Explain-Da-V (which includes textual transformations and aggregations), such a composition is not currently possible. Instead, the explanation Explain-Da-V chose (most valid, see Section 7.4) uses an IMDB Rating attribute to determine the number of genres using a decision tree with a validity of 0.65.[15]

Finally, we note that the performance varies with respect to the different change dimensions. Interestingly, if we only look at vertical removals (Section 5.2), all three baselines have a validity and generalizability score of 1. The reason for that is their ability to discover projections (in their terms applying a drop operation over an attribute). Although it successfully finds these transformations, it lacks the ability to explain the semantics of the attribute removal. Explain-Da-V, although not perfectly valid and generalizable (.95), is more expressive in term of explaining the removal. For example, explaining that an attribute was removed because it contains duplicated information (see Section 5.2). When looking at tuple removal, Explain-Da-V performs much better than the baselines. Since Auto-pipeline does "not consider row-level filtering" [100], we recall the comparison against SQUARES (see Section 8.1.3). Despite its focus on learning a selection predicate, SQUARES is able to resolve cases where a predicate was applied with 0.6 validity



(a) Validity (b) Generalizability (c) Conciseness (d) Concentration

Figure 8: Ablation Study over SDVB datasets.

(Explain-Da-V obtains 0.73 over these changes). This is because SQUARES was not able to resolve removing tuples containing NaN values and duplicate tuples (two cases in the benchmark).

Runtime: In these experiments, excluding timeouts (see Section 8.1.3), finding an explanation using foofah took an average of 4.9 seconds, foofah+ 12.4 seconds, Auto-pipeline* 8.1 seconds, and Explain-Da-V 2.4 seconds. A reason for that difference is that fitting a regressor (linear time complexity) and learning a decision tree (quadratic complexity) are more efficient than search (exponential).

8.3 Explain-Da-V Ablation Study

Figure 8 provides an ablation study of Explain-Da-V. We focus on vertical addition explanations and analyze Explain-Da-V performance without finding the origin (W/O find origin), i.e., using T as a whole to explain a given goal and without the extensions for numeric-to-numeric transformations (W/O extensions). We also analyze the resolved data types by applying Explain-Da-V assuming all types are numeric (All numeric) or all textual (All textual).

As illustrated in Figures 8a and 8b, the full Explain-Da-V provides the most valid and generalizable performance. Adding extensions and finding the origin provide an average performance boost of 30% and 107%, respectively, in terms of validity, while addressing all attribute types as numeric and textual decreases the validity by 35% and 64%, respectively. The NBA version-set demonstrates an interesting case. Since it contains diverse attributes of varying types, without finding origin, Explain-Da-V obtains very low validity and generalizability. Similarly, as mentioned above, since the IRIS version-set mainly consists of numeric attributes, treating all attributes as textual results in very low performance.

[15]The explanation is available in the repository [4].

Examining the explainability (Figures 8c-8d), we observe that while less valid and generalizable, explanations without extensions are more concise and concentrated. If an origin is not found, the explanations are usually less concise and much less concentrated. Finally, numeric explanations are more explainable than textual explanations especially in terms of conciseness. Note that understanding textual explanations also requires domain knowledge in programming while numeric explanations only require basic math. We further provide some additional interesting cases from the experiments in the technical report [88].

9 CONCLUSION

This work laid the groundwork for explaining semantic changes in data versioning. We introduced the notation of explanation and formally defined change dimensions to be explained. Our method, `Explain-Da-V`, uses different types of techniques to resolve and explain changes between a pair of dataset versions. We introduced measures to evaluate explanations and show that `Explain-Da-V` performs better than multiple baselines over an existing adapted benchmark and a newly introduced data versioning benchmark. In future work, we intend to extend `Explain-Da-V` to address additional data types, e.g., dates, and address changes that are triggered by external data such as performing joins and unions.

REFERENCES

[1] 2022. Auto-pipeline benchmark. https://gitlab.com/jwjwyoung/autopipeline-benchmarks.
[2] 2022. AutoPandas Implementation. https://github.com/rbavishi/autopandas.
[3] 2022. Decision Trees. https://scikit-learn.org/stable/modules/tree.html.
[4] 2022. Explanation Example. https://github.com/shraga89/ExplainDaV/blob/main/Explanation_Example.md.
[5] 2022. Featuretools. https://www.featuretools.com/.
[6] 2022. Foofah Extensions (Repository). https://github.com/shraga89/ExplainDaV/blob/main/Code/Ops_added_to_Foofah.py.
[7] 2022. Foofah Implementation. https://github.com/umich-dbgroup/foofah.
[8] 2022. Initial IMDB dataset. https://www.kaggle.com/datasets/harshitshankhdhar/imdb-dataset-of-top-1000-movies-and-tv-shows.
[9] 2022. Initial IRIS dataset. https://www.kaggle.com/uciml/iris.
[10] 2022. Initial NBA dataset. https://www.kaggle.com/justinas/nba-players-data.
[11] 2022. Initial TITANIC dataset. https://www.kaggle.com/competitions/titanic.
[12] 2022. Initial WINE dataset. https://www.kaggle.com/christopheiv/winemagdata130k.
[13] 2022. Lasso Regularization. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html.
[14] 2022. Pandas. https://pandas.pydata.org/.
[15] 2022. Repository. https://github.com/shraga89/ExplainDaV.
[16] 2022. Rigde Regularization. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Rigde.html.
[17] Ziawasch Abedjan, John Morcos, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, and Michael Stonebraker. 2016. DataXFormer: A robust transformation discovery system. In *32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016*. IEEE Computer Society, 1134–1145. https://doi.org/10.1109/ICDE.2016.7498319
[18] Rohan Bavishi, Caroline Lemieux, Roy Fox, Koushik Sen, and Ion Stoica. 2019. AutoPandas: neural-backed generators for program synthesis. *Proc. ACM Program. Lang.* 3, OOPSLA (2019), 168:1–168:27. https://doi.org/10.1145/3360594
[19] Ladjel Bellatreche and Robert Wrembel. 2013. Special issue on: Evolution and versioning in semantic data integration systems. , 57–59 pages.
[20] Anant P. Bhardwaj, Souvik Bhattacherjee, Amit Chavan, Amol Deshpande, Aaron J. Elmore, Samuel Madden, and Aditya G. Parameswaran. 2015. DataHub: Collaborative Data Science & Dataset Version Management at Scale. In *Seventh Biennial Conference on Innovative Data Systems Research, CIDR 2015, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings*. www.cidrdb.org. http://cidrdb.org/cidr2015/Papers/CIDR15_Paper18.pdf
[21] Souvik Bhattacherjee, Amit Chavan, Silu Huang, Amol Deshpande, and Aditya Parameswaran. 2015. Principles of dataset versioning: Exploring the recreation/storage tradeoff. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, Vol. 8. NIH Public Access, 1346.
[22] Tobias Bleifuß, Leon Bornemann, Theodore Johnson, Dmitri V Kalashnikov, Felix Naumann, and Divesh Srivastava. 2018. Exploring change: A new dimension of data analytics. *Proceedings of the VLDB Endowment* 12, 2 (2018), 85–98.
[23] Tobias Bleifuß, Leon Bornemann, Dmitri V Kalashnikov, Felix Naumann, and Divesh Srivastava. 2019. DBChEx: Interactive Exploration of Data and Schema Change. In *CIDR*.
[24] Alex Bogatu, Norman W. Paton, Alvaro A. A. Fernandes, and Martin Koehler. 2019. Towards Automatic Data Format Transformations: Data Wrangling at Scale. *Comput. J.* 62, 7 (2019), 1044–1060. https://doi.org/10.1093/comjnl/bxy118
[25] Leon Bornemann, Tobias Bleifuß, Dmitri Kalashnikov, Felix Naumann, and Divesh Srivastava. 2018. Data change exploration using time series clustering. *Datenbank-Spektrum* 18, 2 (2018), 79–87.
[26] Richard J Brook and Gregory C Arnold. 2018. *Applied regression analysis and experimental design*. CRC Press.
[27] Jason Brownlee. 2022. Data preparation for machine learning.
[28] Douglas Burdick, Ronald Fagin, Phokion G. Kolaitis, Lucian Popa, and Wang-Chiew Tan. 2016. A Declarative Framework for Linking Entities. *ACM Trans. Database Syst.* 41, 3 (2016), 17:1–17:38.
[29] Nadia Burkart and Marco F Huber. 2021. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research* 70 (2021), 245–317.
[30] Hancheng Cao, Vivian Yang, Victor Chen, Yu Jin Lee, Lydia Stone, N'godjigui Junior Diarrassouba, Mark E Whiting, and Michael S Bernstein. 2021. My team will go on: Differentiating high and low viability teams through team interaction. *Proceedings of the ACM on Human-Computer Interaction* 4, CSCW3 (2021), 1–27.
[31] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)* 41, 3 (2009), 1–58.
[32] Adriane Chapman, Elena Simperl, Laura Koesten, George Konstantinidis, Luis-Daniel Ibáñez, Emilia Kacprzak, and Paul Groth. 2020. Dataset search: a survey. *VLDB J.* 29, 1 (2020), 251–272.
[33] Sudarshan S Chawathe and Hector Garcia-Molina. 1997. Meaningful change detection in structured data. *ACM SIGMOD Record* 26, 2 (1997), 26–37.
[34] Sudarshan S Chawathe, Anand Rajaraman, Hector Garcia-Molina, and Jennifer Widom. 1996. Change detection in hierarchically structured information. *Acm Sigmod Record* 25, 2 (1996), 493–504.
[35] James Cheney, Laura Chiticariu, Wang-Chiew Tan, et al. 2009. Provenance in databases: Why, how, and where. *Foundations and Trends® in Databases* 1, 4 (2009), 379–474.
[36] Anton Chernyavskiy, Dmitry Ilvovsky, and Preslav Nakov. 2021. Transformers:"The End of History" for Natural Language Processing?. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 677–693.
[37] Gregory Cobena, Serge Abiteboul, and Amelie Marian. 2002. Detecting changes in XML documents. In *Proceedings 18th International Conference on Data Engineering*. IEEE, 41–52.
[38] Nicole Cruz, Jean Baratgin, Mike Oaksford, and David E Over. 2015. Bayesian reasoning with ifs and ands and ors. *Frontiers in psychology* 6 (2015), 192.
[39] Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeno, Rostislav Petrov, and Preslav Nakov. 2019. Fine-grained analysis of propaganda in news article. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*. 5636–5646.
[40] Anish Das Sarma, Lujun Fang, Nitin Gupta, Alon Halevy, Hongrae Lee, Fei Wu, Reynold Xin, and Cong Yu. 2012. Finding Related Tables. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data* (Scottsdale, Arizona, USA) *(SIGMOD '12)*. Association for Computing Machinery, New York, NY, USA, 817–828. https://doi.org/10.1145/2213836.2213962
[41] Canada Open Data. 2020. https://open.canada.ca/en/open-data
[42] UK Open Data. 2020. https://data.gov.uk/
[43] Boer Deng. 2015. Papers with shorter titles get more citations. *Nature News* 26 (2015).
[44] Dong Deng, Wenbo Tao, Ziawasch Abedjan, Ahmed K. Elmagarmid, Ihab F. Ilyas, Guoliang Li, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2019. Unsupervised String Transformation Learning for Entity Consolidation. In *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*. IEEE, 196–207. https://doi.org/10.1109/ICDE.2019.00026
[45] Jeffrey R Edwards. 2002. Alternatives to difference scores: Polynomial regression and response surface methodology. *Advances in measurement and data analysis* (2002), 350–400.
[46] Kareem El Gebaly, Parag Agrawal, Lukasz Golab, Flip Korn, and Divesh Srivastava. 2014. Interpretable and informative explanations of outcomes. *Proceedings of the VLDB Endowment* 8, 1 (2014), 61–72.
[47] Ahmed K Elmagarmid, Panagiotis G Ipeirotis, and Vassilios S Verykios. 2006. Duplicate record detection: A survey. *IEEE Transactions on knowledge and data engineering* 19, 1 (2006), 1–16.

[48] Peter A Flach and Iztok Savnik. 1999. Database dependency discovery: a machine learning approach. *AI Communications* 12 (3) (1999), 139 – 160. http://content.iospress.com/articles/ai-communications/aic182 Publisher: IOS Press.

[49] Yihan Gao, Silu Huang, and Aditya G. Parameswaran. 2018. Navigating the Data Lake with DATAMARAN: Automatically Extracting Structure from Log Datasets. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein (Eds.). ACM, 943–958. https://doi.org/10.1145/3183713.3183746

[50] Dimitris C Gkikas, Katerina Tzafilkou, Prokopis K Theodoridis, Aristogiannis Garmpis, and Marios C Gkikas. 2022. How do text characteristics impact user engagement in social media posts: Modeling content readability, length, and hashtags number in Facebook. *International Journal of Information Management Data Insights* 2, 1 (2022), 100067.

[51] William R. Harris and Sumit Gulwani. 2011. Spreadsheet table transformations from examples. In *Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2011, San Jose, CA, USA, June 4-8, 2011*, Mary W. Hall and David A. Padua (Eds.). ACM, 317–328. https://doi.org/10.1145/1993498.1993536

[52] Yeye He, Xu Chu, Kris Ganjam, Yudian Zheng, Vivek R. Narasayya, and Surajit Chaudhuri. 2018. Transform-Data-by-Example (TDE): An Extensible Search Engine for Data Transformations. *Proc. VLDB Endow.* 11, 10 (2018), 1165–1177. https://doi.org/10.14778/3231751.3231766

[53] Yeye He, Zhongjun Jin, and Surajit Chaudhuri. 2020. Auto-Transform: Learning-to-Transform by Patterns. *Proc. VLDB Endow.* 13, 11 (2020), 2368–2381. http://www.vldb.org/pvldb/vol13/p2368-he.pdf

[54] Fred Hohman, Kanit Wongsuphasawat, Mary Beth Kery, and Kayur Patel. 2020. Understanding and visualizing data iteration in machine learning. In *Proceedings of the 2020 CHI conference on human factors in computing systems.* 1–13.

[55] The home of the U.S. Government's open data. 2020. https://data.gov/

[56] Silu Huang, Liqi Xu, Jialin Liu, Aaron J Elmore, and Aditya Parameswaran. 2017. ORPHEUSDB: Bolt-on Versioning for Relational Databases. *Proceedings of the VLDB Endowment* 10, 10 (2017).

[57] Zhongjun Jin, Michael R. Anderson, Michael J. Cafarella, and H. V. Jagadish. 2017. Foofah: Transforming Data By Example. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*, Semih Salihoglu, Wenchao Zhou, Rada Chirkova, Jun Yang, and Dan Suciu (Eds.). ACM, 683–698. https://doi.org/10.1145/3035918.3064034

[58] Zhongjun Jin, Michael J. Cafarella, H. V. Jagadish, Sean Kandel, Michael Minar, and Joseph M. Hellerstein. 2019. CLX: Towards verifiable PBE data transformation. In *Advances in Database Technology - 22nd International Conference on Extending Database Technology, EDBT 2019, Lisbon, Portugal, March 26-29, 2019*, Melanie Herschel, Helena Galhardas, Berthold Reinwald, Irini Fundulaki, Carsten Binnig, and Zoi Kaoudi (Eds.). OpenProceedings.org, 265–276. https://doi.org/10.5441/002/edbt.2019.24

[59] Mary Beth Kery, Amber Horvath, and Brad Myers. 2017. Variolite: Supporting Exploratory Programming by Data Scientists. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems.* 1265–1276.

[60] Mary Beth Kery, Bonnie E John, Patrick O'Flaherty, Amber Horvath, and Brad A Myers. 2019. Towards effective foraging by data scientists to find past analysis choices. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems.* 1–13.

[61] Mary Beth Kery, Marissa Radensky, Mahima Arya, Bonnie E John, and Brad A Myers. 2018. The story in the notebook: Exploratory data science using a literate programming tool. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems.* 1–11.

[62] Alexandra Kim, Laks VS Lakshmanan, and Divesh Srivastava. 2020. Summarizing hierarchical multidimensional data. In *2020 IEEE 36th International Conference on Data Engineering (ICDE).* IEEE, 877–888.

[63] Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the Dark Secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP).* 4365–4374.

[64] Max Kuhn and Kjell Johnson. 2019. *Feature engineering and selection: A practical approach for predictive models.* CRC Press.

[65] Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. 2016. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining.* 1675–1684.

[66] Benjamin Letham, Cynthia Rudin, Tyler H McCormick, and David Madigan. 2015. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics* 9, 3 (2015), 1350–1371.

[67] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep entity matching with pre-trained language models. *Proceedings of the VLDB Endowment* 14, 1 (2020), 50–60.

[68] Benjamin Marlin. 2004. *Collaborative filtering: A machine learning perspective.* University of Toronto Toronto.

[69] Zhengjie Miao, Qitian Zeng, Boris Glavic, and Sudeepa Roy. 2019. Going beyond provenance: Explaining query answers with pattern-based counterbalances. In *Proceedings of the 2019 International Conference on Management of Data.* 485–502.

[70] Renée J Miller. 2018. Open data integration. *Proceedings of the VLDB Endowment* 11, 12 (2018), 2130–2139.

[71] Tom Mitchell. 1997. Decision tree learning. *Machine learning* 414 (1997), 52–78.

[72] Heiko Müller, Johann-Christoph Freytag, and Ulf Leser. 2006. Describing differences between databases. In *Proceedings of the 15th ACM international conference on Information and knowledge management.* 612–621.

[73] Menaka Narayanan, Emily Chen, Jeffrey He, Been Kim, Sam Gershman, and Finale Doshi-Velez. 2018. How do humans understand explanations from machine learning systems? an evaluation of the human-interpretability of explanation. *arXiv preprint arXiv:1802.00682* (2018).

[74] Fatemeh Nargesian, Erkang Zhu, Renée J. Miller, Ken Q. Pu, and Patricia C. Arocena. 2019. Data Lake Management: Challenges and Opportunities. *Proc. VLDB Endow.* 12, 12 (aug 2019), 1986–1989. https://doi.org/10.14778/3352063.3352116

[75] Fatemeh Nargesian, Erkang Zhu, Ken Q. Pu, and Renée J. Miller. 2018. Table Union Search on Open Data. *Proc. VLDB Endow.* 11, 7 (mar 2018), 813–825. https://doi.org/10.14778/3192965.3192973

[76] Andrew Nierman and HV Jagadish. 2002. Evaluating Structural Similarity in XML Documents.. In *webdb*, Vol. 2. Citeseer, 61–66.

[77] Pedro Orvalho, Miguel Terra-Neves, Miguel Ventura, Ruben Martins, and Vasco Manquinho. 2020. SQUARES: a SQL synthesizer using query reverse engineering. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2853–2856.

[78] Aslihan Özmen, Mahdi Esmailoghli, and Ziawasch Abedjan. 2021. Combining Programming-by-Example with Transformation Discovery from large Databases. In *Datenbanksysteme für Business, Technologie und Web (BTW 2021), 19. Fachtagung des GI-Fachbereichs „Datenbanken und Informationssysteme" (DBIS), 13.-17. September 2021, Dresden, Germany, Proceedings (LNI)*, Kai-Uwe Sattler, Melanie Herschel, and Wolfgang Lehner (Eds.), Vol. P-311. Gesellschaft für Informatik, Bonn, 313–324. https://doi.org/10.18420/btw2021-16

[79] Thorsten Papenbrock, Jens Ehrlich, Jannik Marten, Tommy Neubert, Jan-Peer Rudolph, Martin Schönberg, Jakob Zwiener, and Felix Naumann. 2015. Functional Dependency Discovery: An Experimental Evaluation of Seven Algorithms. *Proc. VLDB Endow.* 8, 10 (2015), 1082–1093. https://doi.org/10.14778/2794367.2794377

[80] Forough Poursabzi-Sangdeh, Daniel G Goldstein, Jake M Hofman, Jennifer Wortman Wortman Vaughan, and Hanna Wallach. 2021. Manipulating and measuring model interpretability. In *Proceedings of the 2021 CHI conference on human factors in computing systems.* 1–52.

[81] Erhard Rahm and Philip A Bernstein. 2001. A survey of approaches to automatic schema matching. *the VLDB Journal* 10, 4 (2001), 334–350.

[82] John F Roddick. 1995. A survey of schema versioning issues for database systems. *Information and Software Technology* 37, 7 (1995), 383–393.

[83] Pau Rodríguez, Miguel A Bautista, Jordi Gonzalez, and Sergio Escalera. 2018. Beyond one-hot encoding: Lower dimensional target embedding. *Image and Vision Computing* 75 (2018), 21–31.

[84] Sudeepa Roy, Laurel Orr, and Dan Suciu. 2015. Explaining query answers with explanation-ready databases. *Proceedings of the VLDB Endowment* 9, 4 (2015), 348–359.

[85] Maximilian E Schüle, Josef Schmeißer, Thomas Blum, Alfons Kemper, and Thomas Neumann. 2021. TardisDB: Extending SQL to Support Versioning. In *Proceedings of the 2021 International Conference on Management of Data.* 2775–2778.

[86] Vraj Shah, Jonathan Lacanlale, Premanand Kumar, Kevin Yang, and Arun Kumar. 2021. Towards Benchmarking Feature Type Inference for AutoML Platforms. In *Proceedings of the 2021 International Conference on Management of Data.* 1584–1596.

[87] Roee Shraga, Avigdor Gal, and Haggai Roitman. 2020. Adnev: Cross-domain schema matching using deep similarity matrix adjustment and evaluation. *Proceedings of the VLDB Endowment* 13, 9 (2020), 1401–1415.

[88] Roee Shraga and Renée J. Miller. 2022. Explaining Dataset Changes for Semantic Data Versioning with `Explain-Da-V` (Technical Report).

[89] Rishabh Singh. 2016. BlinkFill: Semi-supervised Programming By Example for Syntactic String Transformations. *Proc. VLDB Endow.* 9, 10 (2016), 816–827. https://doi.org/10.14778/2977797.2977807

[90] Rishabh Singh and Sumit Gulwani. 2012. Learning Semantic String Transformations from Examples. *Proc. VLDB Endow.* 5, 8 (2012), 740–751. https://doi.org/10.14778/2212351.2212356

[91] Richard T Snodgrass, Curtis Dyreson, Faiz Currim, Sabah Currim, and Shailesh Joshi. 2008. Validating quicksand: Temporal schema versioning in τXSchema. *Data & Knowledge Engineering* 65, 2 (2008), 223–242.

[92] Charles Sutton, Timothy Hobson, James Geddes, and Rich Caruana. 2018. Data diff: Interpretable, executable summaries of changes in distributions for data wrangling. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 2279–2288.

[93] Christos Thrampoulidis, Samet Oymak, and Babak Hassibi. 2015. Regularized linear regression: A precise analysis of the estimation error. In *Conference on Learning Theory*. PMLR, 1683–1709.

[94] Kai Ming Ting, Sunil Aryal, and Takashi Washio. 2018. Which Outlier Detector Should I use?. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 8–8.

[95] Quoc Trung Tran, Chee-Yong Chan, and Srinivasan Parthasarathy. 2014. Query reverse engineering. *The VLDB Journal* 23, 5 (2014), 721–746.

[96] S Vijayarani, Ms J Ilamathi, Ms Nithya, et al. 2015. Preprocessing techniques for text mining-an overview. *International Journal of Computer Science & Communication Networks* 5, 1 (2015), 7–16.

[97] Xiaolan Wang and Alexandra Meliou. 2019. Explain 3D: explaining disagreements in disjoint datasets. *Proceedings of the VLDB Endowment* 12, 7 (2019).

[98] Yuan Wang, David J DeWitt, and J-Y Cai. 2003. X-Diff: An effective change detection algorithm for XML documents. In *Proceedings 19th international conference on data engineering (Cat. No. 03CH37405)*. IEEE, 519–530.

[99] Cong Yan and Yeye He. 2020. Auto-suggest: Learning-to-recommend data preparation steps using data science notebooks. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1539–1554.

[100] Junwen Yang, Yeye He, and Surajit Chaudhuri. 2021. Auto-pipeline: synthesizing complex data pipelines by-target using reinforcement learning and search. *Proceedings of the VLDB Endowment* 14, 11 (2021), 2563–2575.

[101] Amy X Zhang, Michael Muller, and Dakuo Wang. 2020. How do data science workers collaborate? roles, workflows, and tools. *Proceedings of the ACM on Human-Computer Interaction* 4, CSCW1 (2020), 1–23.

[102] Alice Zheng and Amanda Casari. 2018. *Feature engineering for machine learning: principles and techniques for data scientists.* " O'Reilly Media, Inc.".

[103] Erkang Zhu, Dong Deng, Fatemeh Nargesian, and Renée J Miller. 2019. Josie: Overlap set similarity search for finding joinable tables in data lakes. In *Proceedings of the 2019 International Conference on Management of Data*. 847–864.

[104] Erkang Zhu, Yeye He, and Surajit Chaudhuri. 2017. Auto-Join: Joining Tables by Leveraging Transformations. *Proc. VLDB Endow.* 10, 10 (2017), 1034–1045. https://doi.org/10.14778/3115404.3115409