Datenbank Spektrum (2021) 21:245–249
https://doi.org/10.1007/s13222-021-00392-w

KURZ ERKLÄRT

Kurz erklärt: Measuring Data Changes in Data Engineering and their Impact on Explainability and Algorithm Fairness

Meike Klettke[1] · Adrian Lutsch[1] · Uta Störl[2]

Received: 18 June 2021 / Accepted: 27 September 2021 / Published online: 27 October 2021

Abstract
Data engineering is an integral part of any data science and ML process. It consists of several subtasks that are performed to improve data quality and to transform data into a target format suitable for analysis. The quality and correctness of the data engineering steps is therefore important to ensure the quality of the overall process.
In machine learning processes requirements such as fairness and explainability are essential. The answers to these must also be provided by the data engineering subtasks. In this article, we will show how these can be achieved by logging, monitoring and controlling the data changes in order to evaluate their correctness. However, since data preprocessing algorithms are part of any machine learning pipeline, they must obviously also guarantee that they do not produce data biases.
In this article we will briefly introduce three classes of methods for measuring data changes in data engineering and present which research questions still remain unanswered in this area.

Keywords Data engineering pipelines · Reliability · Explainability · Data bias · Degree of data changes

1 Introduction

Especially when processes are automated, their correctness and reliability is of particular importance. In data driven science and in machine learning knowledge for decisions is derived from data. In most cases, the starting points are big datasets. In data engineering, the features used for data science or machine learning are extracted from these datasets. For large datasets, we need to consider various issues besides the large volume of data, such as heterogeneity, incompleteness, and noise. Most analytics algorithms can only process regular and clean data. Therefore, raw data sets are prepared for their use case.

Data engineering pipelines (such as Talend, IBM InfoSphere DataStage, and Tableau Prep) combine multiple steps to predict data ready-to-be-analysed and to improve data quality. They review, correct and transform data from raw structures into the target formats. During these data preparation steps, several subtasks are solved. Often, these subtasks are assembled into a data science or machine learning pipeline, starting with the data engineering subtasks (data wrangling, different data cleaning methods and data integration) followed by the analysis algorithms. The result of these sequentially executed data engineering tasks is a dataset that differs from the original dataset.

In recent years, the requirements of explainability, reproducibility and fairness have been established for ML approaches and today they are some of the most important research tasks of our time. Algorithm fairness has been discussed widely, for example in [1–6]. In most cases not the machine learning algorithms themselves generate the bias, but the bias is already contained in the data. For this reason the requirements of correctness and ensuring freedom from bias must be applied to the overall process and thus also to data engineering.

There are two different reasons why data can have a bias [7, 8]:

✉ Meike Klettke
meike.klettke@uni-rostock.de

Adrian Lutsch
adrian.lutsch@uni-rostock.de

Uta Störl
uta.stoerl@fernuni-hagen.de

1 University of Rostock, Rostock, Germany

2 University of Hagen, Hagen, Germany

1. The real-world data sets already contain a bias. The algorithms have been trained on biased human decision made in the past.
2. Data bias may result from data engineering steps. Unbalanced selection of data, a combination of datasets, an automatic imputations of missing data and other data cleaning operations may skew the data.

The detection of data bias is particularly important in AI systems because these systems tend to reproduce existing trends and thus also increase the data bias. This behaviour emerges because the results of the procedures cause future biased decisions. If data from these decisions is used to update the ML model, the system forms a vicious cycle of self-reinforcing bias. Therefore, combating data bias and ensuring fairness is an even more important task when it comes to automatically executed procedures [7].

The second reason given above underlines that the requirements explainability and fairness demand that all artificial intelligence subtasks, namely all data engineering steps that prepare the data for the analysis have to be included in this process. It has to be possible to specify, how the data engineering algorithms change the dataset (e.g. the amount of data and the distribution of certain values in the data).

So, after clarifying the importance of the data engineering process for algorithm fairness, in this article we will proceed to establish principled approaches to addressing the underlying issues. In Sect. 2.1, we will introduce how benchmarks are applied for solving this task, in Sect. 2.2 we will introduce methods for logging all data changes of real-data sets and in 2.3 we will sketch which set of information is necessary to foresee data changes for a specific data engineering case. There are still many future tasks in this area, and with these we conclude this article.

2 Methods Determining the Degree of Data Changes

In the following section, we will suggest three different approaches to measuring the data changes introduced during the data preparation:

1. A *black-box approach* based on a gold standard (a *benchmark* which contains input data and the expected correct output data), can be used, applying the data engineering algorithms of the pipeline for the benchmark data and comparing the results against the results that have been defined in the benchmark. This approach does not analyse the algorithms itself but observes the input and output data for one dataset from a benchmark. In case, the results fulfil the requirements (delivers the same output data as defined in the benchmark), then that serves as proof-of-concept and the data engineering pipeline is applied onto the new dataset. This method *only* uses the *data*.
2. It is possible to determine and protocol the *degree and amount of changes* for a given dataset from an application field and a given preprocessing pipeline which consists of a sequence of data preprocessing algorithms. This approach observes the data changes for a dedicated setting and can be seen as a *simulation with a concrete subsequent order of algorithms*. This method uses the *algorithms and the (real) data*.
3. Another approach is an estimation of the expected data changes that is bases solely on an in-depth analysis of the algorithms. This *white-box approach* that inspects the effects of the preprocessing algorithms requires that certain additional information (like contracts) are derived from the algorithms and are stored in a repository of the toolsets. The aim is to foresee the results of each algorithm (sometimes relative to data characteristics, e.g. the number of null values in the dataset) and their sequential combination in a pipeline. This method makes an estimation and uses *only* the *algorithms* for it.

The state-of-the-art for these three different variants is described in the following.

2.1 Usage of Benchmarks

Whereas several *benchmarks for dedicated data engineering subtasks* are already available (e.g. for imputation of missing data [9], for testing of data deduplication [10–13], for outlier detection [14, 15], and for data integration [16]) the developments of *benchmarks for complete data engineering workflows* is still an ongoing task. Currently, there are joint activities in the data engineering community to develop a benchmark for the overall data preprocessing workflow. The availability of such a benchmark (in the best case with data from many different science fields) would be a very valuable opportunity for validating data engineering tool boxes and processes.

Fig. 1 represents how a benchmark, defining a gold standard can be used. The whole data preprocessing pipeline with all its algorithms is treated like a black box. Input data

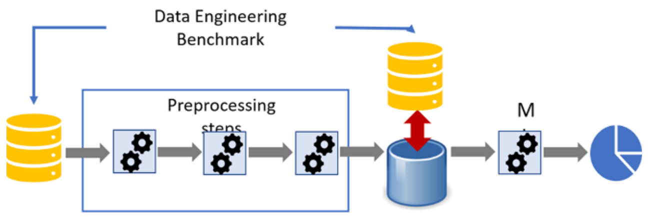

Fig. 1 Application of a Data Engineering Benchmark for validating Data Preprocessing Processes

and the correct output data are predefined by the benchmark. The data engineering process is executed with the input data and the results of the process are compared to the output data (gold standard) in the benchmark. This method provides a very objective opportunity for comparing different pipelines and calculating recall and precision metrics. A disadvantage is that only the predefined datasets (from the standard) can be applied. For real applications (which come with their own data), recall and precision can only be estimated with this method assuming that the data engineering algorithm behave for the real data set in the same way as for the benchmark data.

2.2 Measuring Data Changes

The second option is to measure the *data changes* in a concrete data preparation pipeline *in each step*. This approach does not require a gold standard dataset that provides the correct results. Instead, it uses the real dataset and determines the data changes caused by the different preprocessing algorithms (e.g. imputation of missing values, outlier elimination, deduplication). Fig. 2 sketches this evaluation in the sequential processing of data engineering algorithms.

Data engineering pipelines change the data. To observe the degree of data changes, we have to apply an evaluation on two different levels:

1. The data changes of *each data engineering algorithm* have to be determined. In Fig. 2, the input and output data of each algorithm are visualised by tables. The difference between these tables are the data changes.
2. The data changes of the *whole data preparation pipeline* has to be calculated. This can be achieved by aggregating the data changes by each algorithm. The execution order of the data engineering algorithms influences the result.

We describe both levels in the following. The data changes of each single algorithm can be measured with the following methods:

- *Number of data changes:* by comparing the input and output datasets of an algorithm, the number of changed entries is calculated. For tabular data, this is a *count* function of the number of *added, changed or removed cells in the table* (comparable to the edit operations in the Levenshtein distance). In the case of updated values, we can decide to either
  – calculate the *distance between old and new values* or
  – simply *count* the prevalence of specific changes.

  In case of added or deleted cells, a calculation of differences is not possible because either new or the old value is not available.
- *Differences in data distributions:* Another approach is *measuring and comparing the data distributions* between input and output datasets. It uses distance measures like the earth mover's distance [17] or the total variation distance to compare the distributions of the values inside a column before and after each operation. Depending on the metric, this method can be applied to one-dimensional distributions of single attributes or multi-dimensional, joint distributions of multiple attributes. It was already used for the detection of data engineering methods generating technical data bias [7] or skewing attribute distributions [17].

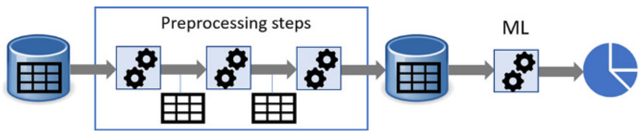

Fig. 2 Estimation of the Data Changes for a Real Dataset in Data Engineering Pipeline

In Fig. 2 we showed one algorithm for each data engineering subpart (like imputation of missing values, outlier detection and elimination or correction, data de-duplication). Indeed, data engineering toolboxes provide several different implementations for each subtask. The analysis of algorithms defined above enables the *comparison of the different implementations* and thus can support the *choice of an algorithm* for a concrete task and concrete dataset and is an opportunity for *reporting* the *degree of data changes* caused by each algorithm.

To determine the total data changes introduced by the whole preparation pipeline, we have to consider that algorithms can amplify or partially undo the data change effects of previous steps. For example, if outlier values are corrected and replaced by mean values, a subsequent duplicate elimination algorithm may treat these added mean values as identical and therefore combine the respective tuples. For this reason, the interaction between algorithms must be considered and procedures are needed to aggregate the change degree across the sequential processes.

The total influence of the data preparation can be measured by aggregating the data changes applied by the single algorithms. A solution for this aggregation that inherently captures the interactions between subsequent algorithms is change lineage tracking. It captures all changes to a dataset by marking changed values. These markings can use different semantics:

- A marking may indicate that a value was changed at least once.
- There may be different markings for every operation in the data engineering pipeline, representing which values were changed by which operations.

- The markings may contain the old value or a distance between old and current value.

In all cases, this method is limited to values inside the final prepared data set. It is comparable to *why* and *how* provenance but restricted to value changes. There is one difference between *supervising data engineering* as introduced in this article and *data provenance*. In data engineering, we assume that in the data cleaning steps data values are changed and we are interested in the degree of changes. In contrast to this, data provenance answers the question why and how certain results are generated and which transformations the data have been passed.

Directly comparing the raw data with the result dataset might be an alternative solution. However, it can be infeasible without lineage information, for example if the preparation pipeline contains aggregation steps preventing direct comparison of single values. We therefore suspect that the step-wise aggregation is the more insightful approach.

2.3 Analysis of Algorithms and Specification of Contracts

If we want to give certain guarantees for the correctness of data engineering processes in advance without knowing the concrete datasets, we have to know the behaviour and characteristics of each of the algorithms. For machine learning approaches, this idea has been introduced in [18] and [19]. In both cases, descriptions for machine learning algorithms like care labels for textiles (or product labels for technical devices like washing machines) have been suggested.

For data engineering tasks we need the same approach. Each data engineering task has its own requirements and data characteristics that must be made explicit (see Fig. 3), these are comparable with the care labels in textiles. The choice of the concrete data engineering algorithms is a match between the requirements that serve as contracts and the concrete algorithms that have to fulfil these contracts.

Whereas the introduction of such contracts is a future task, some data engineering tools (like ETL tools or Tableau Prep) already now provide syntactical tests verifying the applicability of certain algorithms based on syntactical checks

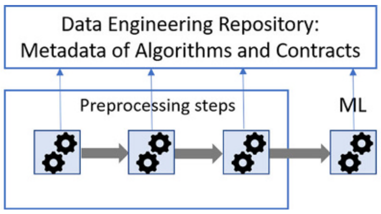
Data Engineering Repository:
Metadata of Algorithms and Contracts
Preprocessing steps
ML

Fig. 3 Defining Characteristics of Data Engineering Pipelines without applying concrete (Test) Datasets

like compatible datatypes which can be seen as a first step in this direction.

For the application in data engineering pipelines, each algorithm could be required to guarantee certain contracts. Issues of such contracts could include:

- *determinism* of the algorithm,
- classification of the changes: *information preserving, extending, or reducing*,
- changes in representations or additionally changes of contents,
- *input requirements*:
  - *algorithm configuration* (parameter setting),
  - *manual human input* of data,
  - usage of additional *external data*,
- *dataset requirements*:
  - *data model* (relational databases, tuples, graphs) and constraints onto the domains (e.g. numerical, categorical, string),
  - does the algorithm require redundancies in the data?
  - necessary *minimal data volume* and data characteristics, and
  - *maximal data volume* (that can efficiently be processed).
- replacement values: how does the algorithm repair values: based on *statistics, rules, external data, additional human correction*?

This list is far from complete. It illustrates the need for a comprehensive guide on algorithm usage and algorithm combination potential for every data engineering algorithm.

Additionally metainformation describing how to sequentially combine the different algorithms is necessary. Especially the specifications, which data engineering algorithms and which machine learning methods can be combined is a task for further investigations and have to be represented in the contracts.

3 Future Work – Open Research Questions

In this article, we have introduced the necessity to guarantee that data engineering algorithms deliver reliable results and do not add bias to data. In investigating this requirement, three different approaches are available: first is based on data only (benchmark data), second is based on data engineering algorithms and real data, and third is based on inspecting the data engineering algorithms only.

For the first approach (see Sect. 2.1), the methodology of using benchmarks for testing certain results is clear and has been proven in many other computer science fields. Here, a task for future work is the development of benchmarks with certain characteristics: (i) containing real data (ii) from different sample applications (iii) which contain different data cleaning and transformation problems.

In the second approach (see Sect. 2.2), (simulating sequences of data engineering steps with real datasets), there are still lots of open research tasks. We need more expressive measures and descriptions for data changes, better methods to aggregate them and effective approaches integrating removed values. Furthermore, we need a better understanding of the interaction between different data engineering steps.

In the third approach (see Sect. 2.3) it is the aim to guarantee certain characteristics of the data engineering without knowing the concrete dataset. Here, we see four open research questions: (i) the definition of the kinds of metadata (contracts) which are necessary, (ii) the determination of the metadata for each algorithms, (iii) based on these contracts the proof which algorithms can be executed one after the other and iv) the combination of these metadata for sequences of algorithms.

To us, it is obvious that data preprocessing algorithms must reliably generate output data with certain characteristics (mainly not introducing a data bias). The application of benchmarks for evaluating data engineering processes (Sect. 2.1), the logging of all data changes through data engineering algorithms (Sect. 2.2) and the estimation of data change by analysing algorithm characteristics (Sect. 2.3) are building blocks to protocol and control data engineering pipelines. The information which is collected in these components can be used to guarantee explainability, checking that data changes do not introduce a data bias and thus indirectly to guarantee algorithm fairness. We believe these task must be future research aims for the whole data engineering community.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Getoor L (2019) Responsible data science. In: SIGMOD
2. Getoor L (2020) Technical perspective: database repair meets algorithmic fairness. SIGMOD Rec 49(1):33. https://doi.org/10.1145/3422648.3422656
3. Salimi B, Howe B, Suciu D (2020) Database repair meets algorithmic fairness. SIGMOD Rec 49(1):34–41
4. Valera I (2019) Fairness in machine learning: from definitions to mechanisms (Keynote LWDA)
5. Venkatasubramanian S (2019) Algorithmic fairness: measures, methods and representations. In: PODS
6. Zweig K (2019) Ein Algorithmus hat kein Taktgefühl: Wo künstliche Intelligenz sich irrt, warum uns das betrifft und was wir dagegen tun können. Heyne, Munich
7. Schelter S, Stoyanovich J (2020) Taming technical bias in machine learning pipelines. IEEE Data Eng Bull 43:39–50. (Special Issue on Interdisciplinary Perspectives on Fairness and Artificial Intelligence Systems)
8. Stoyanovich J, Howe B, Jagadish HV (2020) Responsible data management. Proc VLDB Endow 13(12):3474–3488
9. Lin WC, Tsai CF (2020) Missing value imputation: a review and analysis of the literature. Artif Intell Rev 53(2):1487–1509
10. (2019) Benchmark Datasets for Entity Resolution. https://dbs.uni-leipzig.de/research/projects/object_matching/benchmark_datasets_for_entity_resolution. Accessed: 7 Oct 2021
11. Köpcke H, Thor A, Rahm E (2010) Evaluation of entity resolution approaches on real-world match problems. Proc VLDB Endow 3(1/2):484–493
12. Naumann F, Herschel M (2010) An introduction to duplicate detection. Synth Lect Data Manag. https://doi.org/10.2200/S00262ED1V01Y201003DTM003
13. Saeedi A, Peukert E, Rahm E (2017) Comparative evaluation of distributed clustering schemes for multi-source entity resolution. In: ADBIS
14. Campos GO, Zimek A, Sander J, Campello RJGB, Micenková B, Schubert E, Assent I, Houle ME (2016) On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. Data Min Knowl Disc 30(4):891–927
15. Rayana S (2016) ODDS library. http://odds.cs.stonybrook.edu. Accessed: 7 Oct 2021
16. Poess M, Rabl T, Jacobsen H, Caufield B (2014) TPC-DI: the first industry benchmark for data integration. Proc VLDB Endow 7(13):1367–1378
17. Dasu T, Loh JM (2012) Statistical distortion: consequences of data cleaning. Proc VLDB Endow 5(11):1674–1683
18. Seifert C, Scherzinger S, Wiese L (2019) Towards generating consumer labels for machine learning models. In: Proc. CogMI. IEEE
19. Morik K, Kotthaus H, Heppe L, Heinrich D, Fischer R, Mücke S et al (2021) Yes we care! – Certification for machine learning methods through the Care Label Framework. CoRR. https://arxiv.org/abs/2105.10197