DBKDA 2022

The Fourteenth International Conference on Advances in Databases, Knowledge, and Data Applications

ISBN: 978-1-61208-969-0

May 22nd –26th, 2022

Venice, Italy

DBKDA 2022 Editors

Malcolm Crowe, University of the West of Scotland, UK

Fritz Laux, Reutlingen University, Germany

Andreas Schmidt, University of Applied SciencesKarlsruhe Institute of Technology, Germany

DBKDA 2022

Foreword

The Fourteenth International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA 2022), held between May 22 – 26, 2022, continued a series of international events covering a large spectrum of topics related to advances in fundamentals on databases, evolution of relation between databases and other domains, data base technologies and content processing, as well as specifics in applications domains databases.

Advances in different technologies and domains related to databases triggered substantial improvements for content processing, information indexing, and data, process and knowledge mining. The push came from Web services, artificial intelligence, and agent technologies, as well as from the generalization of the XML adoption.

High-speed communications and computations, large storage capacities, and load-balancing for distributed databases access allow new approaches for content processing with incomplete patterns, advanced ranking algorithms and advanced indexing methods.

Evolution on e-business, ehealth and telemedicine, bioinformatics, finance and marketing, geographical positioning systems put pressure on database communities to push the 'de facto' methods to support new requirements in terms of scalability, privacy, performance, indexing, and heterogeneity of both content and technology.

We take here the opportunity to warmly thank all the members of the DBKDA 2022 Technical Program Committee, as well as the numerous reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and efforts to contribute to DBKDA 2022. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations, and sponsors. We are grateful to the members of the DBKDA 2022 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that DBKDA 2022 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in the fields of databases, knowledge and data applications.

We are convinced that the participants found the event useful and communications very open. We also hope that Venice provided a pleasant environment during the conference and everyone saved some time for exploring this beautiful city

DBKDA 2022 Chairs:

DBKDA 2022 Steering Committee
Fritz Laux, Reutlingen University, Germany
Andreas Schmidt, Karlsruhe Institute of Technology / University of Applied Sciences
Erik Hoel, Esri, USA
Lisa Ehrlinger, Johannes Kepler University Linz, Austria / Software Competence Center Hagenberg GmbH, Austria
Peter Kieseberg, St. Pölten University of Applied Sciences, Austria

DBKDA 2022 Publicity Chairs
Hannah Russell, Universitat Politècnica de València (UPV), Spain
Mar Parra, Universitat Politecnica de Valencia, Spain

DBKDA 2022

Committee

DBKDA 2022 Steering Committee

Fritz Laux, Reutlingen University, Germany
Andreas Schmidt, Karlsruhe Institute of Technology / University of Applied Sciences
Erik Hoel, Esri, USA
Lisa Ehrlinger, Johannes Kepler University Linz, Austria / Software Competence Center Hagenberg GmbH, Austria
Peter Kieseberg, St. Pölten University of Applied Sciences, Austria

DBKDA 2022 Publicity Chairs

Hannah Russell, Universitat Politècnica de València (UPV), Spain
Mar Parra, Universitat Politecnica de Valencia, Spain

DBKDA 2022 Technical Program Committee

Taher Omran Ahmed, College of Applied Sciences, Ibri, Sultanate of Oman / Azzentan University, Libya
Julien Aligon, Institut de Recherche en Informatique de Toulouse (IRIT) | Université Toulouse 1 Capitole, France
Alaa Alomoush, University Malaysia Pahang, Malaysia
AbdulRahman A. Alsewari, Universiti Malaysia Pahang, Malaysia
Emmanuel Andres, Hôpitaux Universitaires de Strasbourg, France
Zeyar Aung, Masdar Institute of Science and Technology, UAE
Gilbert Babin, HEC Montréal, Canada
Jam Jahanzeb Khan Behan, Université libre de Bruxelles (ULB), Belgium / Universidad Politécnica de Cataluña (UPC), Spain
Flavio Bertini, University of Bologna, Italy
Ali Boukehila, University of Annaba, Algeria
Zouhaier Brahmia, University of Sfax, Tunisia
Martine Cadot, LORIA, Nancy, France
Ricardo Campos, Polytechnic Institute of Tomar, Portugal
Sanjay Chaudhary, Ahmedabad University, India
Yung Chang Chi, National Cheng Kung University, Taiwan
Malcolm Crowe, University of the West of Scotland, UK
Monica De Martino, Istituto per la Matematica Applicata e Tecnologie Informatiche "Enrico Magenes" | Consiglio Nazionale delle Ricerche, Italy
Marianna Di Gregorio, University of Salerno, Italy
Anton Dignös, Free University of Bozen-Bolzano, Italy
Ivanna Dronyuk, Lviv Polytechnic National University, Ukraine
Cedric du Mouza, CNAM (Conservatoire National des Arts et Métiers), Paris, France
Lisa Ehrlinger, Johannes Kepler University Linz, Austria / Software Competence Center Hagenberg GmbH, Austria

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

Hybrid Intelligence in Data Privacy Solutions

Marek R. Ogiela
AGH University of Science and Technology
30 Mickiewicza Ave, 30-059 Kraków, Poland
e-mail: mogiela@agh.edu.pl

Lidia Ogiela
AGH University of Science and Technology
30 Mickiewicza Ave, 30-059 Kraków, Poland
e-mail: logiela@agh.edu.pl

***Abstract*—In this paper, we will present the idea of applying the hybrid intelligence technologies in data privacy. This new paradigm joins artificial intelligence (AI) with human intelligence, and allows to support creation of user-oriented cryptographic procedures. Such protocols will apply selected users' preferences in protocols dedicated for increasing data privacy.**

Keywords-Hybrid Intelligence; data privacy; cryptograpic protocols.

I. INTRODUCTION

Modern cryptographic algorithms may be oriented on particular person or group of users. To achieve such feature, it can be implemented using selected AI techniques. There are many contributions presenting security approaches focused on particular users e.g., defining personalized cryptographic protocols [1][2], which can implement personal features or special AI techniques to select unique parameters. Among such techniques we can also find solutions, in which artificial intelligence allows to extract behavioral features or cognitive skills [3][4]. Application of AI procedures allows to define a new area of IT security called cognitive cryptography [5].

Nowadays we can also observe development of new hybrid, i.e., human-AI solutions in security technologies. This means that also in IT security will be possible to introduce hybrid human-AI approaches, which can be focused to guarantee the high security level, and oriented for selected users. Such solutions allow to extend traditional security procedures towards more extensive and optimized analysis of security parameters (or features), and creation of security procedures strongly connected with particular persons. Extensive semantic analysis can be supported by AI solutions, which allow to select the optimal personal parameters or features for created user-oriented security protocols.

Below will be described areas of application of such hybrid procedures, which can be defined for security purposes.

II. HYBRID APPROACHES IN SECURITY PROTOCOLS

The most important areas of application of hybrid security protocols are the following:

- Secure information sharing with privileges
- Knowledge-based authentication protocols
- Visual cryptography
- Personalized behavioral security procedures

In such areas we can define user-oriented security procedures, which involve AI procedures. AI techniques perform optimization tasks, especially important in selection of parameters, or during evaluation of features implemented in security protocols. For example, when we try to define a personalized security protocol we often apply personal features or characteristics. Considering different biometric patterns as well as other specific users features we can evaluate for a particular person a very large feature vector with many personal characteristics. Having such personal record, we can easily select a some of the most distinctive personal features which can later be involved in personalized or user-oriented protocol.

Considering the above mentioned areas of application, we can define the most important features of such protocols as follows.

In secure information sharing, hybrid intelligence can be applied in such manner that user preferences or personal features will have influence on selection of the sharing algorithm, and starting parameters like number of parts, privileges etc. The way of parts distribution can be dependent on AI procedures, which allow to perform division of secret information over several layers.

Knowledge-based authentication protocols can be oriented for particular user or group of persons [6]. In such techniques, expertise knowledge and experiences can be considered, and authentication protocol will be related with thematic visual patterns. Personal features may be related to expertise areas connected with particular user. Having selected the thematic areas AI approaches allow to efficiently select or find visual patterns, which can be presented to user during security procedures.

Visual cryptography is one of the most important areas of application of hybrid intelligence security protocols. Such techniques allow to split visual secret information into several different parts. Usually personal recognition abilities are connected with perception function, and decide when particular user is able to recognize original secret. With such techniques will be possible to establish personal perception thresholds evaluated for different users. Perception levels can

 ISBN: 978-1-61208-969-0

also be dependent on user's knowledge and experiences. In such protocols, knowledge and expectations define human factors, but perception abilities can be evaluated by AI procedures.

Personalized behavioral security procedures allow to define a special type of security protocols, in which different movement feature can be considered [7]. It may contain very simple procedures for personal key generation or more complex protocols oriented for creation behavioral locks. Here we can consider different types of human body movements starting from palm gestures to more complex human motion patterns registered while doing special exercises [1].

In all described areas, the methodology of using hybrid intelligence is the same. To create hybrid human-AI security protocol, firstly the set of personal parameters should be defined. Having defined personal features, the selection of appropriate and unique features can be performed with application of AI methods. The selection of optimal features is usually a very complex task especially in situation when a very large feature vector is available. In such cases, AI approaches allow to quickly select the optimal feature set. Important advantage of application of AI techniques is possibilities to consider constantly changing parameters, which can have different values over the time.

III. CONCLUSIONS

In this paper, we presented possible areas of application of hybrid intelligence techniques used in security protocols. More specifically, the way of application of personal features in advance security solution were described. Additionally, selection of the most important personal parameters can be performed with application of AI procedures.

The most important features of hybrid intelligence security methods are efficiency, and personalization towards application by particular person. Such techniques allow to consider different personal features, and changing parameters associated with users. Hybrid intelligence methods will enrich the cognitive cryptographic approaches defined to join security methods with semantic features or personal parameters [8][9].

ACKNOWLEDGMENT

This work has been supported by the National Science Centre, Poland, under project number DEC-2016/23/B/HS4/00616. This work has been supported by the AGH University of Science and Technology research Grant No 16.16.120.773.

REFERENCES

[1] L. Ogiela, and M. R. Ogiela, "Cognitive security paradigm for cloud computing applications," Concurr. Comput.: Pract. Exp. 32(8), e5316, 2020, doi: 10.1002/cpe.5316.

[2] S. Zapechnikov, "Privacy-Preserving Machine Learning as a Tool for Secure Personalized Information Services," Procedia Computer Science, Vol. 169, 2020, pp. 393-399, doi: 10.1016/j.procs.2020.02.235.

[3] L. Ogiela, "Transformative computing in advanced data analysis processes in the cloud," Inf. Process. Manage. 57(5), 102260, 2020.

[4] M. R. Ogiela, L. Ogiela, and U. Ogiela, "Biometric methods for advanced strategic data sharing protocols,". In: 2015 9th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing IMIS 2015, pp. 179–183, 2015, doi: 10.1109/IMIS.2015.29.

[5] M. R. Ogiela, and U. Ogiela, "Secure information splitting using grammar schemes," New Challenges in Computational Collective Intelligence. Studies in Computational Intelligence, vol. 244, pp. 327–336. Springer, Heidelberg, 2009, doi: 10.1007/978-3-642-03958-4_28.

[6] C. Guan, J. Mou, and Z. Jiang, "Artificial intelligence innovation in education: a twenty-year data-driven historical analysis," Int. J. Innov. Stud. 4(4), 134–147, 2020.

[7] N. Ferguson, and B. Schneier, "Practical Cryptography," Wiley, 2003.

[8] S. J. H. Yang, H. Ogata, T. Matsui, and N.-S. Chen, "Human-centered artificial intelligence in education: seeing the invisible through the visible," Comput. Educ.: Artif. Intell. 2, 100008, 2021.

[9] M. Del Giudice, V. Scuotto, B. Orlando, and M. Mustilli, "Toward the human – Centered approach. A revised model of individual acceptance of AI," Human Resource Management Review, 2021, 100856, doi: 10.1016/j.hrmr.2021.100856.

 ISBN: 978-1-61208-969-0

Context-aware Data Plausibility Check Using Machine Learning

1st Mohaddeseh Basiri
KTH Royal Institute of Technology
Stockholm, Sweden
mbasiri@kth.se

2nd Johannes Himmelbauer
Software Competence Center Hagenberg GmbH
Hagenberg, Austria
johannes.himmelbauer@scch.at

3rd Lisa Ehrlinger
Software Competence Center Hagenberg GmbH
Hagenberg, Austria
lisa.ehrlinger@scch.at

4th Mihhail Matskin
KTH Royal Institute of Technology
Stockholm, Sweden
misha@kth.se

***Abstract*—In the last two decades, computing and storage technologies have experienced enormous advances. Leveraging these recent advances, Artificial Intelligence (AI) is making the leap from traditional classification use cases to automation of complex systems through advanced machine learning and reasoning algorithms. While the literature on AI algorithms and applications of these algorithms in automation is mature, there is a lack of research on trustworthy AI, i.e., how different industries can trust the developed AI modules. AI algorithms are data-driven, i.e., they learn based on the received data, and also act based on the received status data. Then, an initial step in addressing trustworthy AI is investigating the plausibility of the data that is fed to the system. In this work, we study the state-of-the-art data plausibility check approaches. Then, we propose a novel approach that leverages machine learning for an automated data plausibility check. This novel approach is context-aware, i.e., it leverages potential contextual data related to the dataset under investigation for a plausibility check. Performance evaluation results confirm the outstanding performance of the proposed approach in data plausibility check.**

***Index Terms*—Artificial intelligence, Machine learning, Plausibility check, Anomaly detection**

I. Introduction

Due to the rapid development of information technology and manufacturing process, traditional manufacturing enterprises have been transformed to the digital and smart factories [1]. This improvement leads to the emerging complex systems with thousands of components and sub-systems, in which continuous monitoring of these systems is of crucial importance. From the data analytic point of view, this means surveillance of large amounts of time series data in order to ensure the correctness of the data and run data plausibility checks. So, regarding the huge amounts of data, human monitoring of data is not feasible, which conducts us to the automated plausibility check using machine learning and data mining approaches [2].

Data plausibility describes the state when data seems reasonable. Conversely, an anomaly or outlier is a data point that is remarkably different from the remaining data. A possible approach for implementing outlier detection is to run plausibility checks [3]. Rapid and efficient outlier detection is critical for many applications including intrusion detection systems, credit card fraud, sensor events, medical recognition, law enforcement, etc [4]. Although outlier detection is an intensively researched topic in the machine learning and statistics community [5], there are still many open challenges in practice. The first challenge is context dependence. For example, a very high fluctuation rate in a company dataset might be reasonable for a catering service, but not for a construction company. Thus, the decision of whether a data sample seems reasonable (i.e., it is not an outlier) often depends on the context within which it appears. Second, the high dimensionality of the dataset creates difficulties for data plausibility check [6]. Since the number of features increases in a high-dimensional dataset, the amount of data for accurate generalization also raises, which results in data sparsity and scattering. This data sparsity is because of inessential features or irrelevant attributes that hide the correct anomalies. So, anomaly detection is becoming a challenging task by increasing the number of features and attributes in large datasets. In addition to these challenges, there are some inherent issues such as difficulties in the design of threshold between normal and anomalous data, and much noise existence due to incorrect measurements or sensor malfunctioning that may cause the false notifications. On the other hand, data imbalance as the common problem in anomaly detection approaches affects the robustness of models, as very few outlier samples are available.

In order to address the aforementioned challenges, we present a novel context-aware approach for an automated data plausibility check, where there is a lack of research in the literature. In this approach, machine learning techniques are leveraged on top of semantic models, e.g., ontology, and benefited from side information in the datasets. Semantic data models like ontologies [7] facilitate the incorporation of semantic information into the data. The focus of this work is on multivariate outlier detection on the level of records (i.e., samples, rows) instead of single values. In this regard, the main contributions of this work include:

1) Presenting a data plausibility check framework; including test ontology, test data generator, and checkpoint; and their message exchanges,
2) Disclosing three types of tests, to be deployed in the test ontology, executed in the test generator, and used in decision making in the checkpoint module. These tests include:
    a) Inter-feature check, checking features based on

 ISBN: 978-1-61208-969-0

their relations leveraging an Machine Learning (ML) module for prediction of a feature from some related features (list of neighbors is given by the test ontology from training)

b) Intra-feature check (1), checking a feature based on its lags (previous values) using an ML module for prediction based on the lags (number of lags is given by the test ontology from training),

c) Intra-feature check (2), checking a feature leveraging metadata and its long-term statistics (the type of needed metadata and action on them is given by the test ontology)

3) Presenting a comprehensive analysis of the performance of the proposed solution on a propriety dataset and drawing insights and conclusions from the analyses.

The rest of this paper is organized as follows: Section II presents state-of-the-art anomaly detection techniques. Section III presents the data and models used to solve the problem. Section IV describes our solution for solving the problem. Simulation results and discussion are presented in Section V. In Section VI, the findings of this work are presented in a brief but succinct manner.

II. RELATED WORK

Anomaly detection, as the concept of identifying patterns or data points that are significantly different from the expected behavior, has been widely studied. State-of-the-art using anomaly detection algorithms can be categorized as following [8]:

Classification Based: This algorithm strives to discern normal data instances from the abnormal ones in the given dataset space by using a trained model. It is categorized into one-class and multi-class models. In one-class models, a distinguished threshold is learned to label data points outside of this threshold as anomalies instances [9]. In multi-class models, multiple classifiers are trained. A data point is recognized as an anomaly if none of the classifiers can label it as the normal instance [10]. Neural networks, Bayesian networks, support vector machines, and rule-based utilize different classification algorithms to build their classifiers.

Nearest Neighbor Based: In this technique, normal data points are in compact neighborhoods, while anomalous data points are far from their nearest neighbors. This technique needs a distance or similarity measurement between two data points in order to recognize which data points are far from or different from other points. For continuous features, Euclidean distance is used, and for categorical features, a simple matching coefficient is a common option. In multivariate data points, the combination of computed distance for each feature is usually leveraged. The nearest neighbor technique is categorized into two groups regarding how they compute the anomaly score: 1) The distance of a data point to its k^{th} nearest neighbor is used as the anomaly score, e.g., k-nearest neighbor approach [11]. 2) The relative density of each data point is computed as the anomaly score, e.g., Local Outlier Factor (LOF) [12].

Clustering Based: In this algorithm, similar data instances are grouped into clusters. There are three categories of clustering-based anomaly detection techniques. First, techniques that suppose normal data instances belong to a cluster, while abnormal data points do not belong to any cluster, e.g., SNN clustering [13]. Second, algorithms that consider normal data instances are near to the closest cluster centroid, while outliers are far from their closest cluster centroid, e.g., Self-Organizing Maps [14]. Third, those assume normal data instances create large and dense clusters, while anomalous data points create small or scattered clusters, e.g., Cluster-Based Local Outlier Factor (CBLOF) [15].

Statistical: Regarding the basic assumption of statistical anomaly detection techniques, normal data points happen in high probability areas of a stochastic model, while outliers happen in the low probability areas of the stochastic model. In these approaches, a statistical model (usually for normal patterns) is applied to the dataset and then a statistical inference test is utilized to identify whether a data point fits well to this model or not. Regarding the applied test statistic, data instances that there are low probability to be created from the learn model are considered as anomalous data.Parametric and non-parametric techniques are two approaches that can be leveraged to fit a statistical model. Gaussian model based algorithms like Maximum Likelihood Estimation (MLE) [16], regression model based like Auto-regressive Integrated Moving Average (ARIMA) [17], and combination of parametric distribution based algorithms like Expectation Maximization (EM) [18] are instances of parametric techniques. Histogram based such as Intrusion-Detection Expert System (IDES) [19], and kernel function based like parzen windows estimation [20] are samples of non-parametric techniques.

Information Theoretic: In this approach, the information content of the dataset is analyzed. The purpose of this technique is to solve a double optimization problem in order to determine the minimized subset that maximizes the complexity reduction of the dataset, and finally label that subset as the outlier. Entropy and Kolmogorov Complexity [21] are two examples of this category.

Spectral: This technique tries to find a lower-dimensional subspace in such a way that outliers and normal data points are remarkably different. Hence, anomalies can be easily distinguished. Principal Component Analysis (PCA) is used in many techniques in order to project data points into a lower dimensional space [22].

III. DATA AND MODELS FOR EXPERIMENTS

This section sheds light on the data under investigation. Furthermore, it provides details on the pre-processing performed on the received data, and the planned data analytics and verification procedures.

A. Data Collection

The relation of data to AI is as food to the human being. In other words, there is no artificial intelligence in isolation, and any AI approach needs corresponding data for learning.

 ISBN: 978-1-61208-969-0

For this project, we receive the dataset through our industrial partner, from a third-party company. While the data itself is confidential and could not be shared open access on the web, in this section we try to provide insights into the data, in order to make the reader familiar with the approaches that will be presented in the next section.

1) A deep Look into the Dataset: Our dataset contains 18 unique test runs for produced machine parts. Each of these tests has been run for a different period of time, i.e., there are different reported cycles per test.

2) Features available per test: The first dataset (testoverview. csv) provides a comprehensive list of features available per test (out of 18 tests). These features include the type of material used in the experiments, e.g., the oil, and the setting that has been applied in the experiment, e.g., distance between disks. This metadata has been collected to be used for verification of dataset and its reproducibility, as we will see in the next section (Section IV-C).

3) Features available per test cycle: For each of the tests mentioned above, measurements have been done for different periods of time, and a number of features have been recorded per time cycle in the second dataset (tests.csv). In other words, this dataset presents a comprehensive list of features available per time cycle for each test. In contrast to the first dataset, most of the features of the second dataset are unknown to the reader and have not been revealed by the third company to us.

B. Pre-processing of Data

For pre-processing of data, we investigate NaN values and missing entries in the dataset. Then, we start plotting the data to see trends in the results from each test. Figure 1 represents two features of a specific test across time. It is interesting to see that the features represent 3 trends in 3 different phases, including (a) an increasing trend at the start phase (up to 600 cycles, with a return to 50 periodically for the second feature), (b) a semi-constant trend from 600 cycles until the end cycle -600 cycles, and (c) an increasing trend in the last 600 cycles (with a return to 50 periodically for the second feature). In order to see if it is a recurring trend, we investigate the same thing for other tests. The increasing/decreasing trend at the start/end phases and the semi-constant trend in the middle phase are observed in all tests unless one test and this test is excluded from our analysis based on the human expert information, as it does not show the standard behavior.

C. Planned Data Analysis

Figure 2 represents the plausibility check problem and the planned analysis for dealing with this problem. Based on this figure, we receive the data per test per time cycle (as the data pipeline from the bottom of the blue box), and also some metadata per test (as the left data pipeline), and aim at investigating if each test data is plausible or not. The focus of this work is on the design of the plausibility check module and the design of an ontology for the generation of the check data to be used in the plausibility checker module.

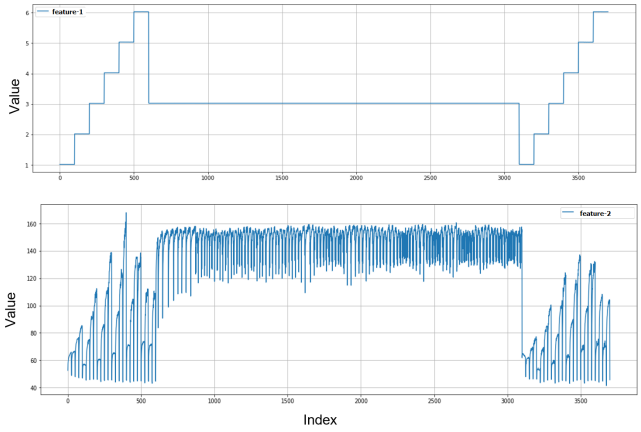


Fig. 1. Description of subset-1 of data versus cycle index

1) Evaluation Metric: In this work, we focus on predicting the test values and comparing them with the real values for detection of a potential anomaly, i.e., performing regression analysis. Regression refers to predictive modeling, and involves predicting a numeric value, and is different from the classification that involves predicting the label of a class of data. In regression analysis, we use Mean Squared Error (MSE), as an error metric designed for evaluating predictions made on regression problems. The MSE metric is derived as the mean or average of the squared differences between real and predicted values, i.e.,: $MSE = \frac{1}{N}\sum_{i=1}^{N}(X[i] - \tilde{X}[i])^2$, in which, $X[i]$ is the i'th real value in the dataset and $\tilde{X}[i]$ is the i'th predicted value. The difference is squared, which has the effect of resulting in a positive error value and inflating or magnifying the large errors.

2) Evaluation Framework: Figure 2 represents the evaluation framework for performance assessment of the proposed plausibility check solution. Based on this figure, we will add two types of error, including constant bias noise and random noise, to the test data per cycle, and will check if the plausibility check module is capable of finding inconsistency in the data.

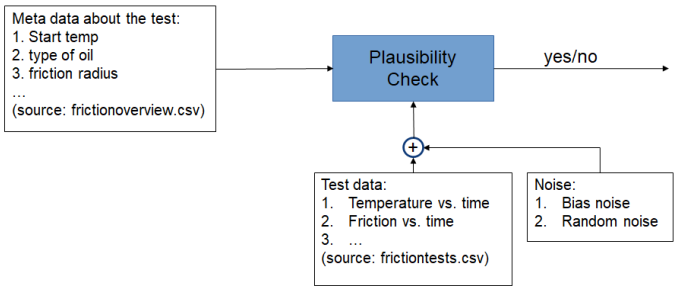


Fig. 2. Planned evaluation framework

IV. THE PROPOSED SOLUTION

This section aims at presenting contributions of the work. Our contributions include the design of a data analytics unit for plausibility check of data. The schema of the proposed

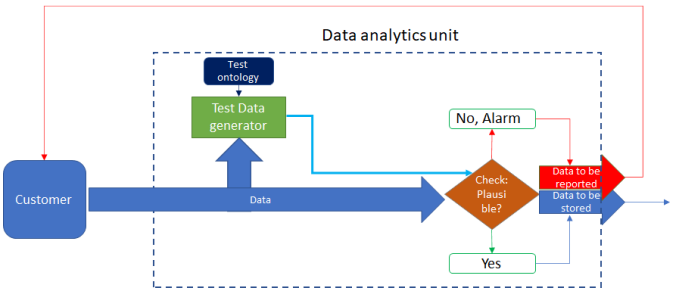


Fig. 3. The proposed solution

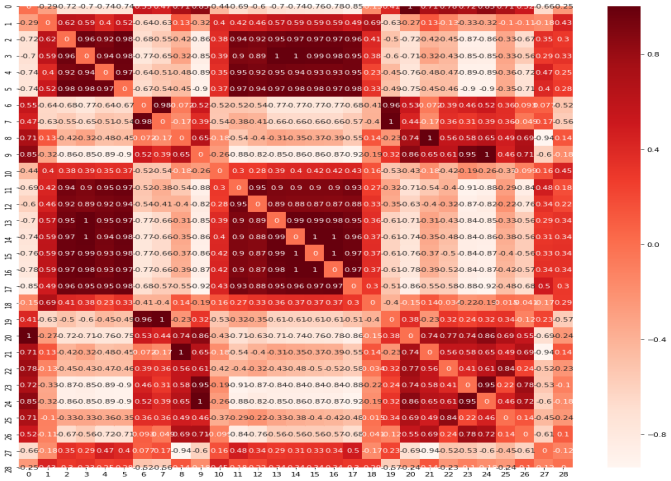


Fig. 4. The correlation matrix after averaging over all available tests

solution has been depicted in Figure 3. This proposed unit includes two novel functions: (a) the test data generator function and (b) the plausibility check function. The former one collects further information about the test and generates checkpoints (contextual data) to be evaluated by the checker function. The checker function compares the checkpoints with the threshold values and makes the plausibility decision. Then, before storing data in the database or actuating based on the received data, the customer can pass the data through the data analytics unit and check whether this data is plausible or not. As we will see in detail of the proposed approaches, the test data generator function includes an intelligent agent for generating the test data.

Implementation of the proposed solution requires contextual data[1] to be collected. Also, the contextual data should be useful in the plausibility check of the dataset. In the following, three ideas are presented for generating contextual data:

1) Cross-correlation between columns of the dataset is used for prediction of the column of interest. The performance of prediction (Mean Squared Error (MSE)) is reported as a property of column of interest for a plausibility check.
2) Prediction of future values of each column based on the previous values of that column and comparison with the received data (Auto-regression). The performance in terms of MSE is used for a plausibility check.
3) Finding rules and statistics for each column based on metadata and configuration available for the test, e.g., type of oil used at the machine part.

A. Design of contextual information for plausibility check: The first solution

In tests.csv dataset, there are 18 unique tests with 29 data columns, unique hash codes, and different cycles. The columns of the dataset could be correlated together. Then, one can use some columns to check the plausibility of other columns.

For testing the hypothesis of mutual correlation between different columns, we consider one unique test and find the correlation between each column with itself and with 28 other columns, by using the built-in correlation function of python. As shown in Figure 4, the correlation results of each test are stored in a matrix of $29 * 29$. The correlation number in each cell $c_{i,j}$ of this matrix is an amount between -1 and 1 and this number states that how much the column i is correlated to the column j. The higher the absolute value of each cell $c_{i,j}$, the more correlated the column i to the column j.

Since the correlations between columns in one test might randomly be high or low, the correlation matrix is calculated for each 18 unique tests, and 18 correlation matrices of $29 * 29$ are obtained. Then, each cell of correlation matrices is averaged over all 18 tests. Figure 4 refers to the result of averaged correlation matrices over 18 tests. This correlation matrix is for the starting phase. Since the behavior of features in the various phases is different, the correlation matrix for the steady-state and ending phase are calculated separately.

As the absolute value of the correlation matrix is of importance, the features with the hottest and coldest colors are more correlated together. As shown in Figure 4, the results confirm the existence of strongly related features for plausibility check of each feature.

Having access to the m most related columns for each column, we can train a machine-learning algorithm to predict the value of feature of interest (FoI) based on the selected features. Here, we select the three most related features for prediction. If the prediction based on the selected features matches the recorded data, there is a low probability of implausibility. If the predicted and recorded values do not match, an alarm could be raised. For deploying this idea, we need an ML agent. Figure 5 depicts the check data generation and decision-making procedures in more detail. In this figure, the FoI is X_1, and the subset of features related to it is X_2. Then, X_2 is fed to the test generator node, and a prediction of X_1 based on X_2 is generated (call it $\tilde{X}_1$). The predicted value, $\tilde{X}_1$ along with X_1 are fed to the comparator node, and from the comparison, the system can carry out the validation process. Finally, the X_1 data will be accepted or an alarm will be triggered. One must note that the test ontology can trigger generating any kind of test data for X_1 based on X_2. For example, after setting the ontology by a human expert, the ML agent in the test generator node takes ontology and customer dataset as inputs. Ontology

[1]Test data related to the dataset to be checked at the plausibility check function

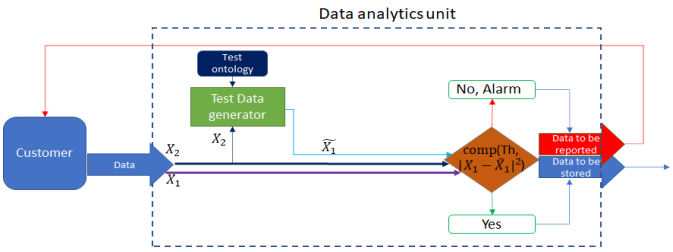


Fig. 5. Feature selection and decision making procedure in more details for the first solution.

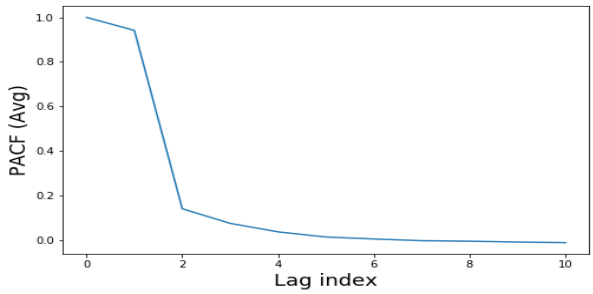


Fig. 6. Average of auto-correlation for feature-1 over different tests in the end phase

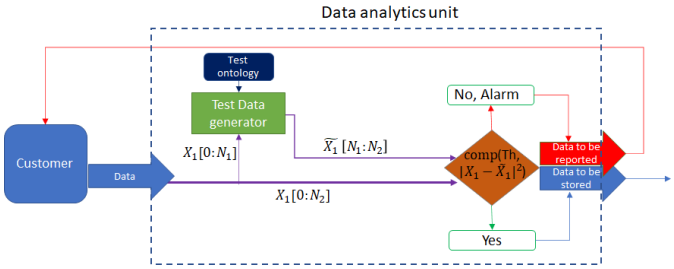


Fig. 7. Feature selection and decision making procedure in more details for the second solution.

determines what contextual information should be collected. In the above example, ML agent understands from the ontology that MSE is required to be collected for the FoI. So, the ML agent, by applying an appropriate algorithm, generates the MSE in the prediction of feature-1 using the three most related features to it. In the decision-making step, this MSE is compared with the ground-truth value. If the value of MSE is less than or equal to the ground-truth value, then the customer data is plausible and can be stored in the database, otherwise, the data is implausible and an alarm is raised.

Towards deploying the ML agent, we need to select an ML algorithm, prepare a train and test dataset, train it over train dataset, and test it over test dataset. To select an ML algorithm, we need to consider some points such as simplicity in usage, scalability, being model-free, explainability, resistance against overfitting and noise, resistance against non-available values in measurements, and working with categorical and continuous values. Regarding these tips, a random forest (RF) algorithm for regression is selected to be implemented in the ML agent. Investigation of the RF algorithm on our dataset for configuration of its parameter, i.e., number of estimator trees, showed us that the best performance, in terms of speed and overfitting, is achieved by 50 trees. Performance of the RF algorithms for plausibility check is investigated in subsection V-A of the next section. Towards using RF algorithm, we train an RF agent based on several tests (out of 18 as described in the previous section), and then test this agent on a test dataset (excluding the training datasets).

B. Design of contextual information for plausibility check: The second solution

Not only does cross-correlation exists between columns of the dataset, but also auto-correlation among values of one column could be considered. It means that one can utilize the previous values of a column to check the plausibility of a specific value in this column.

To see if auto-correlation could be used for the prediction of a feature from its lags, we consider one unique test and find the auto-correlation for each feature of this test. By using auto-correlation, we can find how a value of a feature in time t is related to the previous values of this feature at time $t-1$, $t-2$, $t-3$, ..., $t-n$. Since the values of auto-correlation for a specific feature of one test could randomly be high or low, we repeat auto-correlation for this feature over the 18 tests and average the values of these tests. So, Figure 6 is resulted. Then, among these averaged values, previous m recent values are selected for use in the ML agent. Since the behavior of features in the various phases follows different models, the auto-correlation function is calculated for each phase of a feature separately.

Having access to the previous m recent values of a feature, we can train a machine-learning algorithm to predict the value of FoI at time t based on the previous values of the feature at time $t-1$, $t-2$, ..., and $t-n$. If the prediction value at time t based on the previous m recent values match the recorded data, there is low probability of implausibility, otherwise because of mismatch of prediction data and recorded one, an alarm could be raised. Figure 7 depicts the overall architecture of the second solution in more detail. In this figure, part of $X_1[0:N_2]$, e.g., $X_1[0:N_1]$ in which $N_1 < N_2$, is fed to the test data generator (Note: X_1 is the FoI.). Then, based on the test ontology, e.g., time series forecasting of X_1 using ARIMA, test data for the validity of $X_1[0:N_2]$ will be generated, e.g., $\tilde{X}_1$. Finally, at the comparator node, the real value of X_1 will be compared against $\tilde{X}_1$. Based on this comparison, X_1 data will be accepted or an alarm will be triggered.

Toward deploying an ML agent for the second hypothesis, Random Forest (RF) and ARIMA algorithms are implemented. As mentioned in subsection IV-A, we use the RF algorithm with 50 estimators for our test purpose. For the RF algorithm, the plausibility of each data point is checked based on the 10 lags of the data, i.e., $x[n]$ is checked based on $x[n\text{-}10]$:$x[n\text{-}1]$. For ease of notation, we call this RF algorithm as RF(50,10). For the ARIMA approach, the investigation of parameters on our dataset showed that P=3, Q=I=0, i.e., ARIMA(3,0,0) matches our dataset. Performance of ARIMA

 ISBN: 978-1-61208-969-0

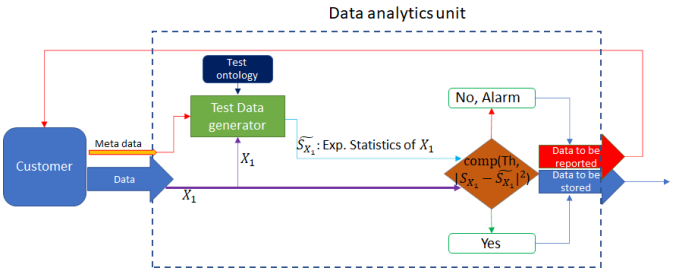


Fig. 8. Feature selection and decision making procedure in more details for the third solution.

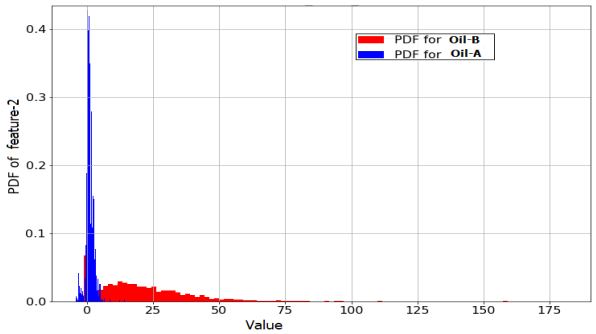


Fig. 9. PDF of *feature-2*

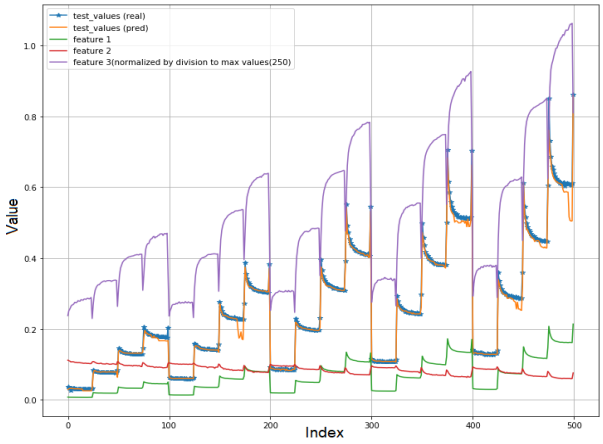


Fig. 10. Testing agent for predicting feature-1 with more details of 3 most related features

and RF algorithms for plausibility check is investigated in subsection V-B of the next section. Towards using ARIMA and RF algorithms, we train the ML agent based on several datasets (out of 18 tests), and then test these agents on a test dataset (excluding training tests).

C. Design of contextual information for plausibility check: The third solution

In the previous sections, we have leveraged the information in the features, either in the FoI or a combination of features, for plausibility check. In other words, the other contextual data gathered by the test maker related to the overall test have not been considered. In this section, we aim at investigating the impact of such contextual data on the statistics of FoI, and the potential application of such connection in plausibility check for the dataset. Figure 8 represents the overall structure of the proposed solution in which, the metadata about X_1, which is the FoI, is also fed to the test data generator along with X_1. Then, based on the test ontology, e.g., partitioning Cumulative Distribution Function (CDF) of X_1 based on states of the metadata, test data for the validity of X_1 will be generated, e.g., $\tilde{S}_{X_1}$. Finally, at the comparator node, the real value of S_{X_1} from received X_1, e.g., the average value of X_1 will be compared against the $\tilde{S}_{X_1}$. Based on this comparison, X_1 data will be accepted or an alarm will be triggered.

In our dataset, there are several contextual information corresponding to each unique test that potentially have impacts on the statistics of features. Examples of such contextual data include type of the *oil* and *separator metal* used in the experiment. Let us focus on oil. The initial hypothesis is that there is a connection between the type of oil used in a test and the statistics of measurements in this test. For example, the min, max, variance, median, mean values of distribution for Oil-A have considerable differences from the ones of Oil-B. Figure 9 shows the statistics for *feature-2*. One can observe that the Probability Density Function (PDF) and Cumulative Distribution Function (CDF) of this feature are different for various oil types. Furthermore, the min and max values of this feature for *type-A* oil differ from *type-B* oil. So, using these explored statistics, we can add some rules to the ontology to discover the implausibility of the data. If the data would be implausible, the related statistics will change in comparison with the normal ones.

We train the metrics of decision-making using statistics of feature-2. If statistics of the test dataset comply with the statistics of the trained dataset, i.e., metrics like min, median, and variance are within the accepted bound found in the training, the decision-maker accepts the test data as plausible. Performance of plausibility check by using statistics of the data is investigated in subsection V-C of the next section.

V. RESULTS AND DISCUSSION

A. Performance test for the first solution

Recall the first proposed solution in Figure 5. In this solution, the test ontology mandates predicting FoI for validity check based on the three most-related features. It also proposes MSE as the prediction analysis metric. Then, the three most related features to the FoI are fed to the test data generator and are used for predicting the FoI. Figure 10 shows the performance test results such that the prediction values are fitted well with the real values of FoI (here, feature-1).

In this figure, along with the test data and predicted data, the three most related features to feature-1 can be seen as well. One can observe that these three features have almost either direct or inverse (because of negative values of correlation) relationship with the feature of interest (feature-1). The above tests have been repeated for the steady phase and ending phase, and the same behavior has been almost observed for tests in these phases. We aim at leveraging the proposed ML agent

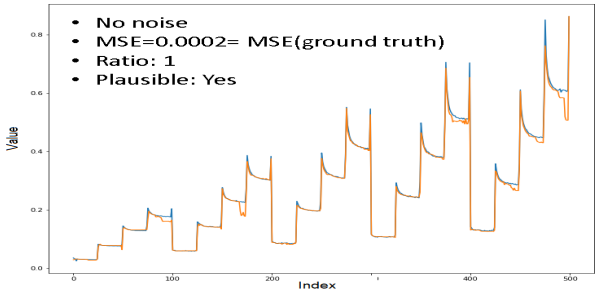


Fig. 11. Testing agent for predicting feature-1 based on 3 most related features. MSE=0.0002= MSE ground truth. (Blue: real data , Orange: predicted data)

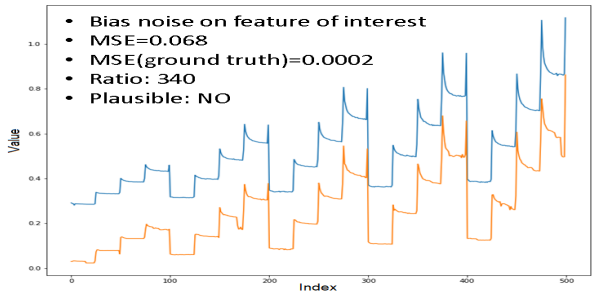


Fig. 12. Testing agent for predicting feature-1 based on 3 most related features. Bias noise on the FoI, MSE=0.068 (340x ground truth). (Blue: real data , Orange: predicted data)

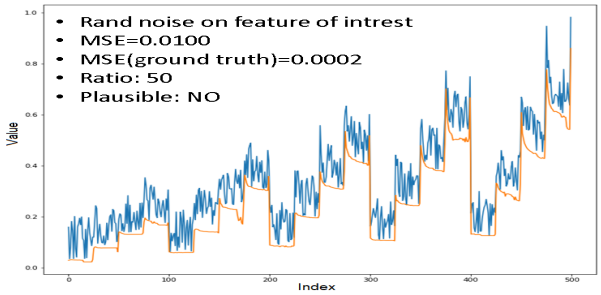


Fig. 13. Testing agent for predicting feature-1 based on 3 most related feature. Random noise on the FoI, MSE =0.01 (50 times higher than the ground truth). (Blue: real data , Orange: predicted data)

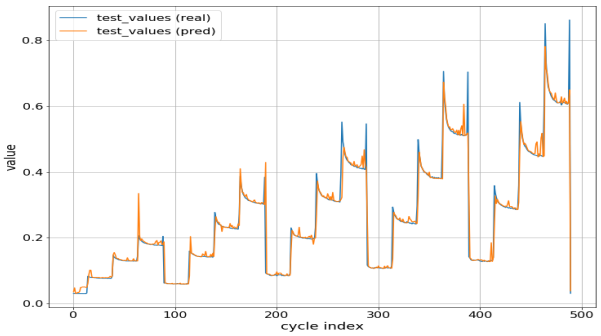


Fig. 14. Testing agent for predicting feature-1 using auto-correlation and random forest.

for carrying plausibility checks out. So, seven test cases are presented based on applying bias measurement errors, and random measurement errors to the column of interest, and most related columns. The first plausibility check is related to the state that there is no noise in the data. As shown in Figure 11, the plausibility of data has been confirmed. In the second plausibility test, bias noise is added to the feature of interest (feature-1). From results of Figure 12 are observed that the predicted values are not the same as real values. So, the ML agent can detect the error on the data and conclude the implausibility of data. The third plausibility test is related to the adding bias noise to the least related feature. In the fourth plausibility check, bias noise is added to the most related feature. The same plausibility tests are done by adding random noise on the FoI, least related feature, and most related feature. The result of adding random noise on the feature-1 is shown in Figure 13. Table I summarizes the results of these seven plausibility tests for the first solution.

B. Performance test for the second solution

Recall the second proposed solution in Figure 7. In this solution, the test ontology mandates predicting FoI for validity check based on its lags. It also proposes MSE as the prediction analysis metric. Then, the lags of FoI are fed to the test data generator, and are used for predicting the FoI. Figure 14 shows the performance test result using random forest for predicting feature-1 (FoI). In the random forest algorithm, we used 10 recent values of feature-1 for prediction. Figure 15 depicts the performance test result for predicting feature-1 using auto-correlation and ARIMA. In our implementation, ARIMA works with three recent values of feature-1. Both Figure 14 and Figure 15 confirm that the prediction values fit well with the real values of feature-1. We do the performance test for the steady phase and ending phase of feature-1 using random forest and ARIMA algorithms and the results for these phases also follow the same trend.

For plausibility check using the auto-correlation contextual data, we apply bias measurement errors and random measurement errors to the FoI, and examine if the proposed solution can assess the incorrectness of data. Towards this end, we leverage the FoI's forecasting results using ARIMA and random forest methods. From Figure 16, Figure 17, Figure 18, and Figure 19 with random and bias noises, one can observe that the predicted values are not the same as real values of FoI. So, the ML agent can detect the error on the data and

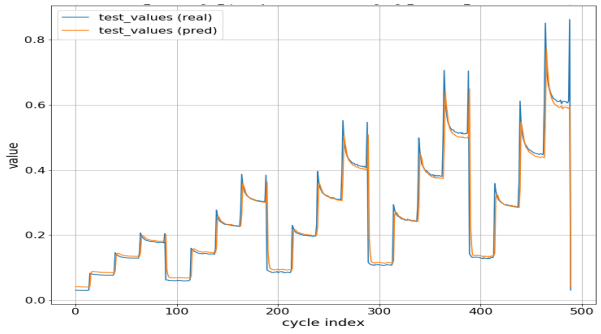


Fig. 15. Testing agent for predicting feature-1 using auto-correlation and ARIMA.

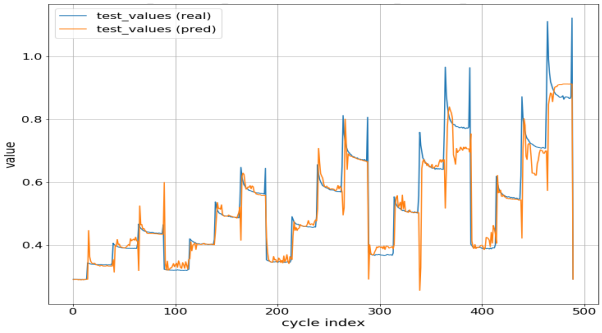


Fig. 16. Plausibility check for predicting feature-1 using auto-correlation, Random forest, and bias noise.

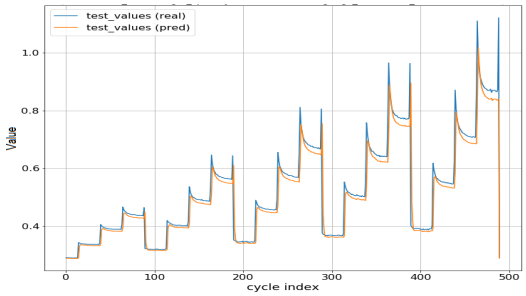


Fig. 17. Plausibility check for predicting feature-1 using auto-correlation, ARIMA, and bias noise.

conclude the implausibility of the data. Table II summarizes the results of plausibility tests for the second solution.

C. Performance test for the third solution

Recall the third proposed solution in Figure 8. In this solution, the test ontology collects metadata about FoI for validity check. Then, the past values of this feature are fed to the test data generator, and are used for extraction of statistics of this feature, and predicting the validity of the feature based on the extracted statistics. Here, we focus on the oil data and try to partition the pdf of FoI based on the type of oil used in the experiment. Figure 20 represents the partitioned pdf of the feature-2 (FoI) based on the type of oil used in the experiment. One observes the same trend from the test data and train data when there is no noise added to data (plausible test dataset).

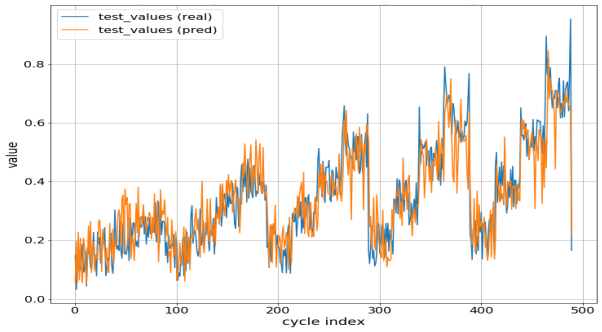


Fig. 18. Plausibility check for predicting feature-1 using auto-correlation, random forest, and random noise.

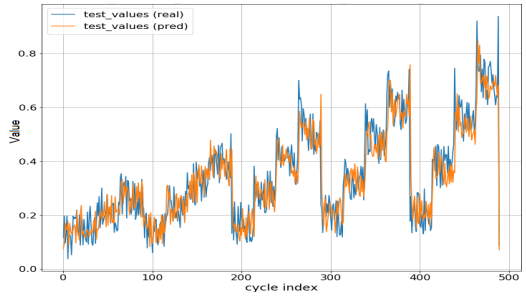


Fig. 19. Plausibility check for predicting feature-1 using auto-correlation, ARIMA, and random noise.

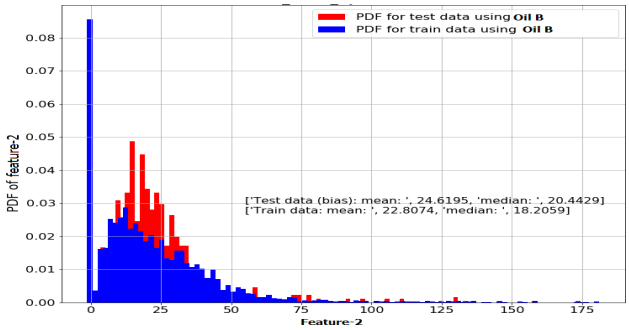


Fig. 20. Comparison of PDF of FoI in two tests

In this section, we apply bias noise and random noise on the test data to check if our designed solution can detect the implausible data. Figure 21 and Figure 22 show the results of performance analysis for bias and random noise respectively. One observes in Figure 21 that adding the noise to the test data (red one) clearly shifts the plot to the right. Figure 22 represents the dataset with random noise. One observes that the noise has changed the shape of the pdf, e.g., the mean and median have changed.

D. Discussion

In Table I, the results of plausibility test for the first solution (subsection V-B) have been summarized. Table II summarizes the performance results for the second solution (subsection V-B). Table III summarizes the results of figures 20, 21, and 22 in subsection V-B.

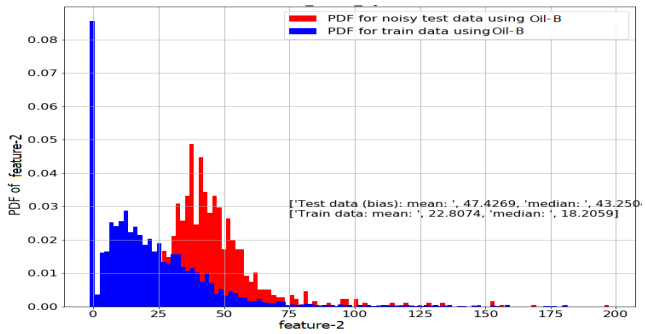


Fig. 21. Comparison of PDF of FoI with and w/o bias noise

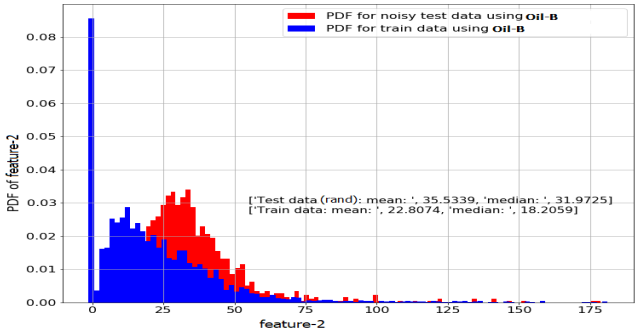


Fig. 22. Comparison of PDF of FoI with and w/o random noise

TABLE I
SUMMARY OF PLAUSIBILITY CHECK USING SOLUTION 1

Test description	**MSE**	**MSE$_{ratio}$: $\frac{MSE}{MSE_{true}}$**	**Check**: MSE$_{ratio}$ < Ratio$_{th}$; Ratio$_{th}$ = 1.5
True data	0.0002	1	Y
Bias error on column of interest (feature-1)	0.068	360	N
Bias error on least related feature	0033	16.5	N
Bias error on most related feature	0.0420	210	N
Random error on feature of interest (feature-1)	0.010	50	N
Random error on least related feature	0.0012	6	N
Random error on most related feature	0.0068	34	N

TABLE III
SUMMARY OF PLAUSIBILITY CHECK USING SOLUTION 3

Test description	**Mean**	**Mean-ratio**	**Median**	**Median-ratio**	**Plaus. check**
Train data (base measurement)	22.8	1	18.2	1	-
Test data w/o noise	24.6	1.08	20.4	1.12	Y
Test data with bias error	47.42	2.08	43.4	2.38	N
Test data with random error	35.8	1.57	31.7	1.74	N

From Table I, it is clear that the plausibility check solution, which is powered by the prediction of FoI based on the most related features, performs well against the bias noise. In other words, when a constant value, i.e., a measurement error, is added to the reading of a sensor, the plausibility check module can easily detect that data is inconsistent with the past learning (from 16.5 to 360 times more MSE has been reported). For the random noise, when the amount of the added noise to the data could vary, the performance is lower than the bias noise, but still completely acceptable (from 6 to 50 times more MSE has been reported). For example, one observes that the plausibility test has been shown 6 times more MSE in the prediction of FoI when random noise on the least relevant feature to the FoI has been added. Furthermore, Table II showed that the second solution (using RF) is not vulnerable to the random noise, and it performs equivalently for the bias and random noises (7 times more MSE in prediction of FoI). In the same time, we observe that the ARIMA has a poor performance as an ML agent for this solution, and it misses the alarm for the test-case with bias noise on the the FoI (the corresponding MSE-ratio is 1.125, which is lower than the threshold value, i.e., 1.5). Finally, the third approach shows a weaker performance than the previous ones (around two times more MSE has been reported). One must note that the stronger performance of the first approach and relatively the second approach is achieved at the cost of further computing required for them. In other words, there is a hidden reliability-complexity tradeoff here, where going from solution 1 to 3, complexity is reduced and the probability of error in plausibility check is increased.

VI. Conclusions

In this work, we investigated data plausibility checks for a given dataset from a smart factory. Towards this end, a data analytics unit, consisting of a contextual data generation function (which generates checkpoints based on a given ontology) and a plausibility check function (which works based on the designed checkpoints), was proposed. For the implementation of the first function, we have investigated three machine learning approaches that leverage auto-correlation in each feature, correlation between features, and hidden statistics of each feature for generating the checkpoints. Performance evaluation results indicated the outstanding performance of the proposed scheme in the detection of noisy data. The main concluding remarks of this work include: (i) This study indicated that each feature of the dataset, or a collection of features, could be used without any other data for plausibility check leveraging machine learning. (ii) Metadata about the test, including conditions in which the test has been carried out, could be an important part of the design of the plausibility check. (iii) Checking of plausibility for a dataset that may contain random noise on some features (or some cycles) is much harder than checking the presence of static noise on the data. (iv) Performances of different checkpoint generation functions (using different ML approaches) are not the same. The ones based on the investigation of each cycle of the test, solutions 1 and 2, are more complex and provide a better distinction between noisy and healthy data. While the third solution is a lightweight solution with a lower reliability performance.

 ISBN: 978-1-61208-969-0

TABLE II
SUMMARY OF PLAUSIBILITY CHECK USING SOLUTION 2

Test description	**MSE in prediction of FoI using itself by ARIMA**	**$\text{MSE}_{\text{ratio}}$ for ARIMA: $\frac{\text{MSE}}{\text{MSE}_{\text{true-AR}}}$**	**Plausibility for ARIMA: $\text{MSE}_{\text{ratio}} < \text{Ratio}_{\text{th}}$; $\text{Ratio}_{\text{th}} = 1.5$**	**MSE in prediction of FoI using itself by RF**	**$\text{MSE}_{\text{ratio}}$ for RF: $\frac{\text{MSE}}{\text{MSE}_{\text{true-RF}}}$**	**Plausibility for RF: $\text{MSE}_{\text{ratio}} < \text{Ratio}_{\text{th}}$; $\text{Ratio}_{\text{th}} = 1.5$**
True data (feature-1)	$0.0024 = \text{MSE}_{\text{true-AR}}$	1	Y	$0.0012 = \text{MSE}_{\text{true-RF}}$	1	Y
Bias error on feature of interest (feature-1)	0.0027	1.125	Y	0.0046	7.8	N
Random error on feature of interest (feature-1)	0.0063	2.8	N	0.0088	7.3	N

ACKNOWLEDGEMENT

The research reported in this paper has been partially funded by BMK, BMDW, the State of Upper Austria in the frame of the COMET Programme managed by FFG and by the EC H2020 project *"DataCloud: Enabling the Big Data Pipeline Lifecycle on the Computing Continuum"* (Grant nr. 101016835).

REFERENCES

[1] V. Q. Nguyen, L. Van Ma, and J. Kim, "Lstm-based anomaly detection on big data for smart factory monitoring," *Journal of Digital Contents Society*, vol. 19, no. 4, pp. 789–799, 2018.

[2] N. Laptev, S. Amizadeh, and I. Flint, "Generic and scalable framework for automated time-series anomaly detection," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1939–1947.

[3] S. So, J. Petit, and D. Starobinski, "Physical layer plausibility checks for misbehavior detection in v2x networks," in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, 2019, pp. 84–93.

[4] C. C. Aggarwal, "Outlier analysis," in *Data mining*. Springer, 2015, pp. 237–263.

[5] C. C. Aggarwal and S. Sathe, *Outlier ensembles: An introduction*. Springer, 2017.

[6] S. Thudumu, P. Branch, J. Jin, and J. J. Singh, "A comprehensive survey of anomaly detection techniques for high dimensional big data," *Journal of Big Data*, vol. 7, no. 1, pp. 1–30, 2020.

[7] L. Ehrlinger and W. Wöß, "Towards a definition of knowledge graphs." *SEMANTiCS (Posters, Demos, SuCCESS)*, vol. 48, pp. 1–4, 2016.

[8] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[9] V. Roth, "Kernel fisher discriminants for outlier detection," *Neural computation*, vol. 18, no. 4, pp. 942–960, 2006.

[10] D. Barbara, N. Wu, and S. Jajodia, "Detecting novel network intrusions using bayes estimators," in *Proceedings of the 2001 SIAM International Conference on Data Mining*. SIAM, 2001, pp. 1–17.

[11] Y. Song, J. Huang, D. Zhou, H. Zha, and C. L. Giles, "Iknn: Informative k-nearest neighbor pattern classification," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2007, pp. 248–264.

[12] A. H. Abuzaid, "Identifying density-based local outliers in medical multivariate circular data," *Statistics in Medicine*, vol. 39, no. 21, pp. 2793–2798, 2020.

[13] G. Moreira, M. Y. Santos, J. M. Pires, and J. Galvão, "Understanding the snn input parameters and how they affect the clustering results," *International Journal of Data Warehousing and Mining (IJDWM)*, vol. 11, no. 3, pp. 26–48, 2015.

[14] R. Smith, A. Bivens, M. Embrechts, C. Palagiri, and B. Szymanski, "Clustering approaches for anomaly based intrusion detection," *Proceedings of intelligent engineering systems through artificial neural networks*, vol. 9, 2002.

[15] S. Ali, G. Wang, R. L. Cottrell, and T. Anwar, "Detecting anomalies from end-to-end internet performance measurements (pinger) using cluster based local outlier factor," in *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*. IEEE, 2017, pp. 982–989.

[16] F. W. Scholz, "Maximum likelihood estimation," *Wiley StatsRef: Statistics Reference Online*, 2014.

[17] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "A comparison of arima and lstm in forecasting time series," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2018, pp. 1394–1401.

[18] G. E. Box and G. C. Tiao, "A bayesian approach to some outlier problems," *Biometrika*, vol. 55, no. 1, pp. 119–129, 1968.

[19] Y. Xiao, C. Xing, T. Zhang, and Z. Zhao, "An intrusion detection model based on feature reduction and convolutional neural networks," *IEEE Access*, vol. 7, pp. 42 210–42 219, 2019.

[20] E. Parzen, "On estimation of a probability density function and mode," *The annals of mathematical statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[21] P. Vitányi, "How incomputable is kolmogorov complexity?" *Entropy*, vol. 22, no. 4, p. 408, 2020.

[22] L. Parra, G. Deco, and S. Miesbach, "Statistical independence and novelty detection with information preserving nonlinear maps," *Neural Computation*, vol. 8, no. 2, pp. 260–269, 1996.

 ISBN: 978-1-61208-969-0

Conflict-free Replicated Hypergraphs

Aruna Bansal
AN & SK School of Information Technology
Indian Institute of Technology Delhi
Delhi, India
email: aruna.bansal@cse.iitd.ac.in

***Abstract*—Hypergraphical structures provide a natural mathematical way to represent the richness of diversified data and complex relationships in hierarchical, relational, navigational, and semi-structured settings, e.g., bibliographic paper submissions, mHealth, and social media applications. These applications operate in distributed environments with a requirement of availability while coping with high network latencies. Replication is the commonly used approach to achieve a high degree of availability, facilitating local query processing. However, replication requires expensive (and often infeasible) concurrency control to ensure consistency. In this work, we specify the *well-formed* Hypergraphs as a Conflict-free Replicated Data Type (HgCRDT), which is a commutative replication-based approach expressed in terms of two 2P-Sets, the latter comprising mutable hyperedges.**

Index Terms*—*Hypergraphs; semi-structured data; complex relationships; well-formed structures; higher-ordered relationships; eventual consistency; data replication; conflict-free replicated data types.

I. Introduction

Hypergraphs are generalized graphs denoted as a pair (N, E) where N is a set of vertices, and E is a set of hyperedges which are arbitrary nonempty subsets of N [1]. Hypergraphs are interesting mathematical structures with application in databases [1]–[4]. Hypergraphs are better-suited than graphs and relational databases to represent complex relationships between hierarchical, navigational, semi-structured data and metadata found in various applications [5]–[8]. *Complex relationships* connect and represent multiple entities and/or relations (to formulate higher-order relations) describing a group of similar entities or a structure. We can find the natural occurrence of complex relationships represented via hyperedges in several applications and data sets, including co-authorship, co-citation, social networks, email networks [6], biological processes [9] [10], and patients' medical history [5]. Higher-order relations can be easily accommodated in hypergraphical structures by employing *higher-order hyperedges* to connect other hyperedges. Traditional data models for data and complex relationships are optimized for particular types of queries and data; for instance, a tabular representation of relational databases is optimized for structured data and relational algebraic queries; and a graph representation of graph databases is optimized for data stored in nodes and edges, as well as navigation and neighborhood queries. Hypergraphs can be used to combine the properties of various data models to cope with the semi-structured, hierarchical, complex, higher-order relationships inherent in such data.

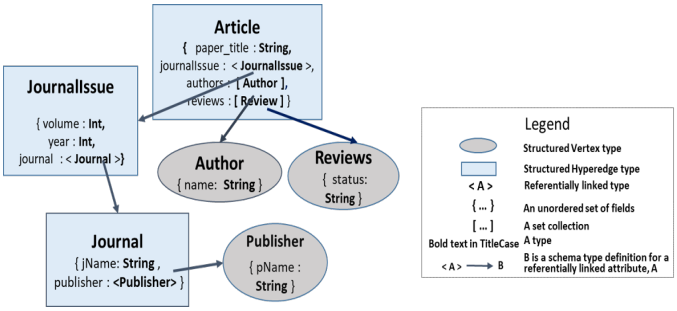


Figure 1. Example: a hypergraph structure capturing an article relationship.

Consider a motivating scenario in which prospective authors submit papers to a journal that are subjected to reviews before being accepted for publication in a journal. The end result is a published paper or a journal article viewed as a relationship between the authors, a collection of reviews, and a journal issue. Journals usually have multiple *issues*; therefore, a journal issue has its publishing year and volume and links the journal (that further relates to a publisher). Note the italicised *issue* refers to journal issue. The paper is an implicit and essential part of the article. Figure 1 depicts various entities (i.e., `Author`, `Reviews`, and `Publisher`) and relationships (i.e., `Article`, `JournalIssue`, and `Journal`) of this scenario in a structured hypergraph via structured vertices and structured hyperedges, respectively, each with a set of fields as their components that are initialized as null.

In this type of real-world scenario, the submission, review, and publication process of conferences/journals are often carried out at many distant domain sites, with each site notifying the other sites of the article's revised status for the next stage. These sites may span over a large geographical area bearing diverse network connections. Therefore, information availability is highly required along with network latencies. Since replication provides availability at the expense of *strong consistency* between the copies, that further necessitate synchronization [11]. Therefore, a weaker notion of consistency is required to ensure the consistency of replicated copies.

We are familiar with consensus algorithm [12] [13] that resolves conflicts between updates, however, at the expense of high reconciliation cost. Here, *Conflict-free Replicated Data Types* or *CRDTs* is a reasonable choice for maintaining consistency in highly dynamic environments [14] [15]. CRDTs address the twin requirements of availability of data (for

 ISBN: 978-1-61208-969-0

efficient local query processing) and operation under network partitioning without requiring complicated concurrency control mechanisms while offering *strong eventual consistency* [16].

Contributions: The purpose of this paper is to discuss how to distribute higher-ordered hypergraphs to multiple replicas in a scalable manner where rejoining replicated copies of hypergraphs from distributed databases is possible without any loss of information. We believe that our work's novelty adds a new dimension to use the rich hypergraphs in distributed settings. Other data distribution challenges, such as security and privacy, are beyond the scope of this paper. We omit specifics about our ongoing implementation to emphasize the suitability of our proposed hypergraphs to be used with conventional CRDTs. On the other hand, the implementation introduces a database paradigm for modeling, storing, retrieving, distributing, and encoding our envisioned hypergraphs.

We aim to leverage the novelty of CRDTs with hypergraph structures and semantics to provide consistent updating and propagation of hypergraphical information across multiple replicas in the previously-stated distributed settings while also ensuring data availability and network latency. Therefore, we extend the existing portfolio of CRDTs [14], [15], [17]–[19] to embrace *well-formed higher-order recursively-defined mutable hypergraph* as a new CRDT: Hypergraph CRDT or HgCRDT. While previous constructions of a CRDT in others have been graphical, hierarchical, list-oriented, key-value based, we believe this is the first instance of well-formed higher-order hypergraphs. In particular, the hyperedges are *mutable* (discussed in Section III-A), in that the set of atoms they connect can be changed. We propose a hypergraph *atom*, a logical term to refer to a vertex or a hyperedge. The mutability of the atoms within a CRDT merits special attention. The hyperedges allow the nesting of hyperedges and are built on references, making their members independent.

In the rest of this paper, we overview background and some related work in Section II. In Section III, we introduce hypergraphs in HgCRDT, and the HgCRDT approach. Next, in Section IV, we present the specification of the HgCRDT that incorporates query, add, remove and modify operations on hypergraphical atoms. A proof-of-correctness showing how concurrent processes meet convergence conditions (essential for eventual consistency) is given in Section V. Finally, in Section VII, we summarise our contributions as well as potential research directions.

II. Background and Related Work

This section will begin by briefly introducing hypergraphs and related work. After that, we will discuss the background of CRDTs and the research aligned with this paper.

A. Hypergraphs

A **hypergraph** is a generalized graph where hyperedges connect more than two vertices. A traditional hypergraph is specified as a pair (N, E) where N is a set of vertices, and E is a set of hyperedges, which are arbitrary nonempty subsets of N as given in [1].

Hypergraphs have been studied since 1980 by various researchers. A few significant work includes: [1] expresses the relational database schemes as hyperedges for ensuring certain degrees of acyclicity (such as α-acyclicity, β-acyclicity, and γ-acyclicity); [20] introduces *Hypernode model* based on nested graphs; [7] proposes *GROOVY*, an object-oriented database model formalized using hypergraphs; [21] proposes a framework for mapping a generic hierarchical/network/relational db model into another using hypergraph; and [22] introduces a schema-oriented graph model with properties and labels using hyper-nodes and hyper-edges.

Existing research on hypergraphs in distributed settings includes, *HyperX* [23] (a scalable hypergraph framework which works in distributed graph settings converting hypergraphs into graphs using a layer built atop Spark), and *Trinity* [24] (a hypergraph database and computation platform over distributed memory cloud).

B. CRDTs

The CRDTs manage distributed replicas of *mutable data* with minimal synchronization and without using complex concurrency control protocols [15]. In CRDTs, different replicas of a data structure can be locally read and written to, replicating the data/operations asynchronously at the distributed locations. The approach applies to data type representations in which the operations performed are conflict-free while ensuring *strong eventual consistency* [16], allowing local modification to the data and then immediately returning to computation.

The CRDTs are designed to work in an underlying *reliable causally-ordered broadcast communication protocol*, in which a source replica (the replica that sends its update information to other replicas) delivers its messages to each downstream replica (the recipient replica) exactly once in an order consistent with happened-before [15]. Maintaining *causal consistency* via a reliable delivery mechanism helps to further reduce inconsistencies between replicas by restricting the operations seen in possibly different orders at the replicas to only *concurrent operations*. Therefore, the same replica can simultaneously send and receive different or redundant messages.

A CRDT specifies an internal data structure representation (called the *payload*), and an collection of *interface operations*, comprising the *query operations* which interrogate the state of the data object and return a value, and the *update operations* which change the internal state of the data object. Both query and update operations may specify *preconditions* that must be satisfied for them to be invoked. Query operations can be performed purely locally, without any need for synchronization and communication with other replicas. Update operations do not return any value, and involve two phases- first, the source site *prepares the parameters* for the updates to be performed at the various replicas; then these changes are *effected* at the various replicas atomically, *immediately* at the source site, and *asynchronously* propagated to the other sites. Usually,

the effect-phase parameters sent to all downstream sites are identical to those at the source phase.

CRDT implementations are classified into Convergent Replicated Data Type (CvRDT) and Operation-based Commutative Replicated Data Type (CmRDT). CvRDTs is a passive replication approach where all necessary information that needs to be replicated is captured by a state which is further transmitted to all the replicas. On the other hand, CmRDTs is an active replication approach where an update operation occurs at the source replica and then is replicated to all the downstream replicas by transmitting operations and performed locally. The eventual transmission of the entire state in CvRDTs may be costly for large data structures. In contrast, operation-based CRDTs transmit only the update operations, which are typically small. However, the communication infrastructure used for CmRDTs must ensure that all operations on a replica are delivered to the other replicas, without duplication, but in any order.

The portfolio of CRDTs [14] [15] includes a variety of interesting data types such as counters, registers, sets, graphs, lists [18] [19], and maps [25]. Of particular interest are the operations-based 2P2P Graph CRDTs [14] [15], since their payload consists of two 2P-Sets for adding and removing vertices and edges where the edge sets are dependent on the vertex sets.

The existing work in the direction of higher-order CRDT includes Riak [25], a distributed NoSQL key-value data store that defines maps as a CvRDT; JSON data structure [26] that composes lists, maps, and registers to embed JSON data types as a CRDT; Logoot [18] that uses a sparse non-mutable *n*-ary tree to nest ordered lists; and higher-order patterns [27]. Causal Graphs [28] illustrates a hierarchical graph-oriented CRDT that represents ordered trees into Causal Graphs. Furthermore, Delta CRDT [29] [30] discusses CRDTs that encode CRDTs using delta mutations of state-based CRDTs. Deltas are temporarily stored in a buffer instead of propagating the entire state to the remote replicas.

An instance of CRDTs employed in databases is the use of *SU_Sets*, a CRDT to handle RDF-Graphs and the SPARQL 1.1 Update operations [31] [32]. The underlying CRDT used in that work is an Operations-based OR-Set of database triples. While the insert and delete operations involve *sets* of elements, these are of a pre-defined atomic element type, in contrast to our higher-order hypergraphs where the set of a hyperedge may include hyperedges belonging to the same hypergraph. More interesting is the insert-delete operation, which uses a *multiset* of mappings when preparing sets of triples to delete and insert into the database.

III. Representation of HgCRDTs

A. Hypergraphs in HgCRDT

We use hypergraphs in HgCRDT, where a (higher-order) *hypergraph* is a collection of *schematic & typed vertices* V and *hyperedges* H. We propose a term hypergraph *atoms* to abstractly refer to the schematic typed vertices and hyperedges. Vertices are assumed to be primitive and represent entities, whereas a *directed* hyperedge is of the form $he(U)$, which connects a *set* of *atoms* U. We formally define hypergraphs in HgCRDT as follows:

Definition 1 (Hypergraphs in HgCRDT). *A hypergraph G in HgCRDT is defined as a collection of hypergraph objects composed of (V, H), where*

- *V is a finite set of vertex objects defined as: $V = \{v_1, v_2, ..., v_n\}$, $n \geq 0$, with each $v_i \in V$ containing only scalar data (such as String, Int, Float, Boolean); and*
- *H is a finite set of hyperedge objects used to represent a relationship, and is defined as: $H = \{he_1, he_2, ..., he_m\}$, $m \geq 0$. Each hyperedge object $he \in H$ connects a finite set of atoms specified as $U = \{u_1, u_2, ..., u_k\}, k \geq 0$, where each $u_i \in (V \cup H)$. The hyperedge he is added to H when the following constraints satisfies avoiding any cycle and self-loop for every $u_i \in he.U$, where $u_i \in H$:*
  1) $u_i \neq he$,
  2) $\forall\ he' \in H$:
     a) $u_i \notin he'.U$
     b) $\forall\ he'' \in he'.U :\ he'' \notin he$ □

Hyperedges are a non-trivial data type, supporting relational structure, hierarchical data, and higher-order relations. A hypergraph is **well-founded** if for every new hyperedge he to be added in H, every atom $u \in he.U$ must exist in $(V \cup H)$. As a consequence, a hyperedge cannot appear within its own set. Notably, we treat atoms as typed objects with a unique implicit identity, avoiding the need to store the entire hyperedge where hyperedge members are themselves (independent) objects. As shown in Figure 2, a few hypergraph objects where, e.g., the independent objects for a journal issue, a set of authors, and a set of reviews are all referentially tied to an article hyperedge object using the implicit object it. Additionally, it aids in the formation of *well-formed* and *acyclic* hypergraphical structures. A hyperedge is said to be **well-formed** when added to H if on adding atom $u \in he.U$, u does not form any cycle and self-loop. Significantly, *acyclic structures* facilitate query optimization and minimize query response time. Hyperedges are **mutable**, in that we permit the set of atoms to be modified. Moreover, hypergraphs are particularly well-suited for replication since a hyperedge's projection containing only some of its set's atoms is still a hyperedge.

Further, similar to vertices that represent entities, each hyperedge retains a set of *internal attributes* (usually of scalar types) that define the properties of a relationship. The internal attributes are different than the referential attributes (i.e., $he.U$) used to define the parts of a relationships based on which the relationship exists. In Figure 1, `paper_title` in `Article`, and `JName` in `Journal` hyperedges are internal attributes. Furthermore, two hyperedges with the same referential set U and carrying the same value are *not* the same. The implicit object ids, the hyperedge type, and the internal attributes of the hyperedges make the hyperedges different. In this paper,

we skip the internal attributes of vertices and hyperedges, and emphasize on using only the referential attributes U in a hyperedge.

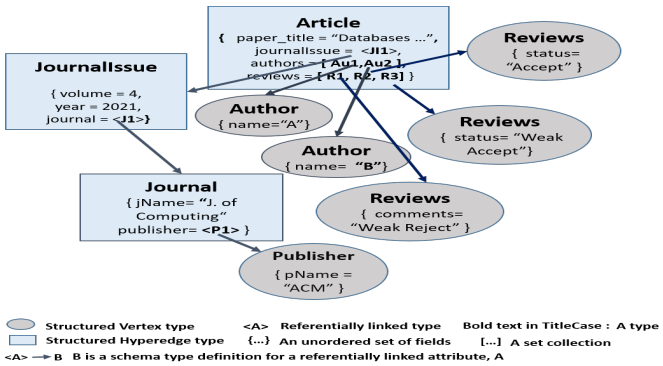


Figure 2. A few hypergraph objects generated for the hypergraph structure of Figure 1.

B. Hypergraph as a CRDT (HgCRDT)

Due to the anticipated size of hypergraphs (for instance, a conference may include sub-conferences, workshops, and a few journals; or a journal/conference may receive a large volume of submissions [33]), we prefer the operations-based commutative (CmRDT) approach over a state-based (CvRDT) or a delta state-based approach as in [30], as transferring the hypergraph state between replicas and merging would be prohibitively expensive.

The communication model of the HgCRDT is similar to that of the CRDTs [15], in that operations are sent in an ordered causal fashion. We employ two 2-phase sets (2P-Sets [15]), i.e., those where elements can be removed after addition, but cannot be reintroduced, as the payload for our hypergraph data type- one each for its vertices V and hyperedges H. Other variants of CRDT Sets are possible, such as OR-Sets, though the commutativity properties need to be carefully verified for each such choice.

We are already familiar with the existing 2P2P graph-based CRDT described in [14]. To facilitate the adoption of our technique, we use the template provided by the 2P2P-Graph specification for hypergraphs. Hypergraphs are generalized graphs dealing with more complex structures than graphs, hierarchies, and maps. The richness of our hypergraphs makes our work different compared to the existing 2P2P-Graph CRDT. Our proposed hypergraph specification uses two tombstone sets (or remove sets: VR, HR) to represent the 2P-Sets, which relaxes in some instances the requirement for a causal order of delivery, and thus permits some additional asynchrony.

Also, note that in hypergraphs, vertices are the base case for atoms (which also include hyperedges) and that hyperedges relate the atoms of a *set* to each other. The novelty of this work lies in this treatment of such well-founded recursive hypergraphical structures. Another novelty is that the set incident on a hyperedge is itself *mutable* 2P-Set. Hyperedges have the following form, in which object references are used to store the set rather than the complete hyperedge itself (in the implementation).

$$he(\textbf{mutable}\ atom\ set\ U)$$

Consequently, hyperedges are mutable, as we may add and remove atoms incident on the hyperedge. The use of tombstone sets allows deletion of an atom from a set; however, since the atoms are implemented as typed objects having their implicit identity, the atoms persist across such modifications. The usage of implicit object identities explains why traditional CRDT models like Key-Value pairs and maps are not suitable for encoding hypergraphs, even after some transformation.

IV. Specification of HgCRDTs

The HgCRDT specification comprises a list of local query operations and global commutative update operations to *add, remove* individual or a set of atoms and *modify* hyperedges. Vertex *modify* is a trivial operation, and therefore, we ignore it in this report. Note that the notion of sources and downstream sites is not statically fixed.

In continuation to our previous example, Figure 3 illustrates a distribution scenario using the HgCRDT framework involving its update operations to capture the journal article's submission, review, and publication processes among three distinct copies. Each update operation begun at a source replica is propagated to subsequent downstream replicas. Note that vertices and hyperedges are introduced to the system in case of no earlier existence, and the outcome is an article with the associated entities and other relationships, as seen in Figure 2. The operation delays affect the payload of a replica in case any other operation needs its prior delivery. As seen in Figure, replica 1 initiates add operations for two vertices for authors A and B and an article hyperedge, which is then shared with replicas 2 and 3. Similarly, replica 2 adds three review vertices. Meanwhile, replica 3 has a vertex for the publisher and two hyperedges for the journal and *issue*. Now, adding the set of reviews, and journal issue at the respective replica (i.e., replicas 2 and 3), necessitates modifying the article hyperedge. Sharing the change operation by both the replicas leads to a concurrent arrival at replica 1. However, this issue is resolved according to the causal order of delivery that reflects both the changes in the article. To add, we assume that a special *issue* of the same journal is introduced to which the article is best suited. Note that the previous instance of the journal issue hyperedge cannot be deleted due to its reliance on the article hyperedge. Thus, the modification of the new *issue* to the article hyperedge enables the deletion of the first *issue*.

Next, we describe the HgCRDT specification that contains a few keywords- *initial* specifies initial values of payload sets at every replica; *let* marks non-mutating statements; *query* and *Update* indicate a non-mutable, and a mutable operations respectively; and *pre* specifies preconditions that must be satisfied for an operation to be invoked. Each update operation has two phases: *prepare at source*, and *effect at downstream*. The initial phase illustrates that an argument is locally prepared by the *source* replica to be delivered to *downstream* replicas.

 ISBN: 978-1-61208-969-0

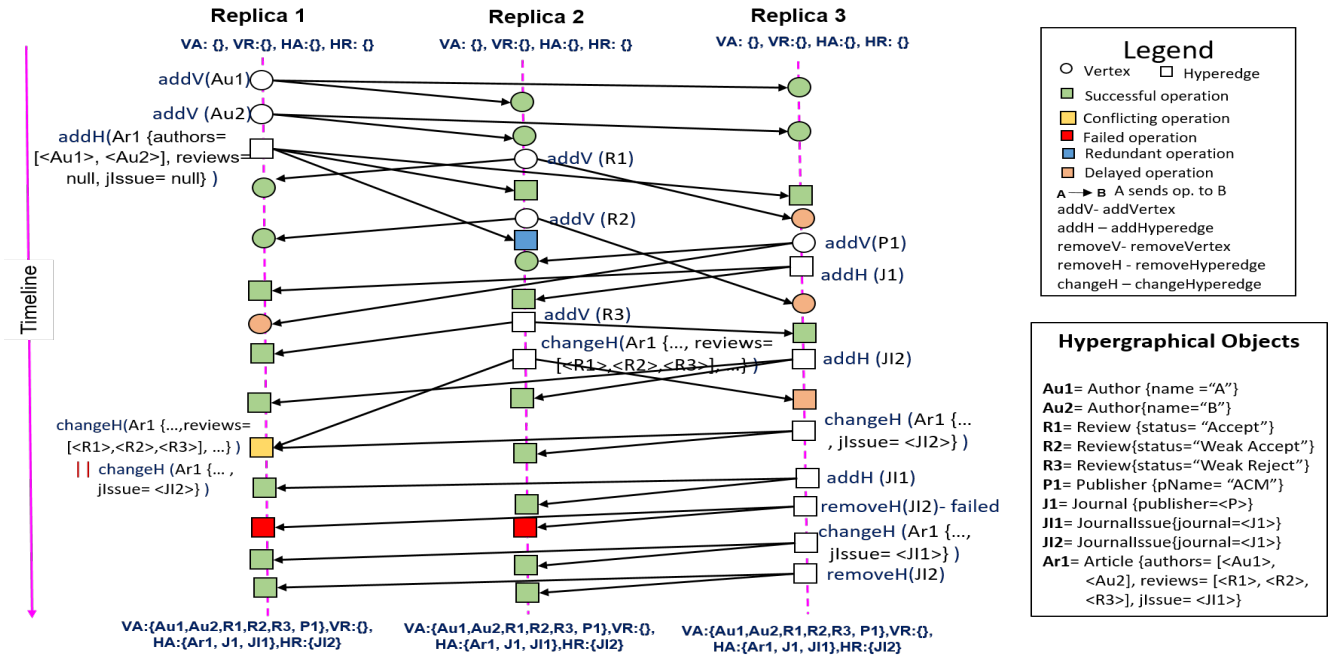


Figure 3. Example: A distribution of HgCRDT operations between three replicas capturing the formation of a journal article relationship (from the paper submission to its publication). Only objects are displayed here, with limited scalar values.

A downstream replica executes the later phase that atomically and asynchronously uses the received argument prepared by the source replica.

a) ***Query operations (Figure 4):****:* The *payload* initializes the local and mutating state of a replica. In HgCRDT, the payload consists of four sets: VA, VR, HA, and HR for adding and removing vertices and hyperedges. The query operations are performed locally at each replica. These operations provide the extensional observational criteria for identifying/distinguishing between the state of the mutable object (hypergraph in this case).

lookupAtom checks for the presence of an atom, whether a vertex (*lookupVertex*) or a hyperedge (*lookupHyperedge*), as the case may be, in the hypergraph. The *lookupAtom* operation is lifted to sets of atoms using conjunction. In the *lookupHyperedge* query, the precondition checks for the existence of all atoms in the set. Since we permit the set to be mutable, the payload sets HA, HR only contain reference-based structures for the hyperedges, and the set is accessed by dereferencing. *within* operation checks that the given hyperedge should be acyclic. Therefore, it recursively checks if a given atom appears within a given hyperedge.

b) ***Update operations****:* are global operations that are defined using the novelty of the CRDT approach. As in, the *source replica* initiates operation and prepares the update information to send to *downstream replicas*. The operation is then *effected* immediately at the source, and if the parameter is non-trivial, also sent asynchronously but reliably to the downstream locations, where it is affected atomically. Causal delivery reduces the need for commutativity to only the concurrent operations, handling the dependency of the hyperedge

$\triangleright$ VA : *vertex add set*, VR : *vertex remove set*,
HA : *hyperedge add set*, HR : *hyperedge remove set*
payload set $VA,\ VR,\ HA,\ HR$
initial ϕ, ϕ, ϕ, ϕ

query $lookupAtom$ (atom a) : boolean b
if a is a vertex: $lookupVertex\ (a)$
otherwise if a is a hyperedge: $lookupHyperedge\ (a)$

query $lookupAtomSet$ (atom set S) : boolean b
let $b = \left(\bigwedge_{\forall u \in\ S} lookupAtom(u) \right)$

query $lookupVertex$ (vertex v) : boolean b
let $b = (v \in (VA \setminus VR))$

query $lookupHyperedge$ (hyperedge $he(U)$): boolean b
let $b = (lookupAtomSet(U) \wedge\ he(U) \in (HA \setminus HR))$

query $within$ (atom a, hyperedge $he(U)$): boolean b
let $b = \begin{cases} true & \text{if } a = he(U) \vee \\ & \quad a \in U \vee \exists\, he(U')\ s.t. \\ & \quad lookupHyperedge(he(U')) \\ & \quad \wedge\ a \in U' \\ & \quad \wedge\ within(he(U'), he(U)) \\ false & \text{otherwise} \end{cases}$

Figure 4. Query Operations

update $addAtom$ (atom a)
  if a is a vertex: $addVertex\ (a)$
  otherwise: $addHyperedge\ (a)$

update $addVertex$ (vertex v)
  prepare at source (v)
  effect at downstream (v)
    $VA := VA \cup \{v\}$

update $addVertexSet$ (vertex set X)
  prepare at source (X)
  effect at downstream (X)
    $\forall v \in X\ : (VA := VA \cup \{v\})$

$\triangleright\ he(atom\ set\ U)$
update $addHyperedge$ (hyperedge $he(U)$)
  prepare at source $(he(U))$
    pre $lookupAtomSet(U)$
  effect at downstream $(he(U))$
    pre $lookupAtomSet(U)$
    $HA := HA \cup \{he(U)\}$

Figure 5. Add Operations

update $removeAtom$ (atom a)
  if a is a vertex: $removeVertex(a)$
  otherwise : $removeHyperedge(a)$

update $removeVertex$ (vertex v)
  prepare at source (v)
    pre $lookupVertex(v)$
      $\wedge\ \forall\ (he\{U\} \in (HA \setminus HR)) :$
      $\neg U.lookupVertex(v)$
  effect at downstream (v)
    pre $addVertex(v)$ delivered
    $VR := VR \cup \{v\}$

update $removeHyperedge$ (hyperedge $he(U)$)
  prepare at source $(he(U))$
    pre $lookupHyperedge(he(U)) \wedge$
      $\forall\ (he\{U'\} \in (HA \setminus HR)) :$
      $\neg U'.lookupHyperedge(he(U))$
  effect at downstream $(he(U))$
    pre $addHyperedge(he(U))$ delivered $\wedge$
      $\forall (he\{U'\} \in (HA \setminus HR)) :$
      $\neg U'.lookupHyperedge(he(U))$
    $HR := HR \cup \{he(U)\}$

Figure 6. Remove Operations

2P-Set on the vertex 2P-Set.

Add Operations (Figure 5): The *addAtom* operation adds a vertex or a hyperedge, depending on the kind of atom specified. Adding a set of hyperedges can be realized by iterating the *addHyperedge* operation. Note that when adding a hyperedge, all atoms in its set must exist, and thus the corresponding add operations for all these atoms must have been delivered earlier.

$\triangleright\ he(\mathbf{mutable}\ atom\ set\ U)$
update $changeHyperedge$ (hyperedge $he(U)$,
    atom set S^+, S^-)
  prepare at source ($he(U)$, atom set S^+, S^-)
    pre $lookupAtomSet(S^+)$
      $\wedge\ U.lookupAtomSet(S^-)$
      $\wedge\ lookupHyperedge(he(U))$
      $\wedge\ \forall(x \in S^+) :\ \neg\ within(\ x,\ he(U))$
  effect at downstream ($he(U)$, atom set S^+, S^-
    pre $addHyperedge(he(U))$ delivered
      $\wedge\ lookupAtomSet(S^+)$
      $\wedge\ \forall(x \in S^+) :\ \neg\ within(\ x,\ he(U))$
    $\forall(x \in S^-) :\ U.removeAtom(x);$
    $\forall(x \in S^+) :\ U.addAtom(x);$

Figure 7. Modify Operation

Remove Operations (Figure 6): We can only delete an atom incident on a hyperedge after the hyperedge itself has been removed. Note that deleting an atom (whether vertex or hyperedge) requires that it should not be incident on *any* hyperedge (should not be in the set of any hyperedge). Thus the precondition ensures that it cannot possibly appear within any higher-order hyperedge. We do not present here the *remove* operations lifted to a set of atoms.

Modify Operations (Figure 7): It is always possible to modify a hyperedge in a hypergraph by deleting the existing edge and replacing it with the modified edge. It requires ensuring that any atoms present (recursively) within the new set of the new hyperedge must already exist (and must not be the hyperedge itself).

However, since hyperedges are complex structures, this implementation is expensive. Instead, we specify the modification of a hyperedge by the addition or removal of atoms in a set via *changeHyperedge* operation. Note that we now require a set to *itself* be *mutable* 2P2P-Set. The vertices and hyperedges in the sets $U.V, U.H$ are respectively subsets of the two 2P-Sets V, H of the global hypergraph object. By global hypergraph, we mean the replica's state consisting of payload sets, irrespective of any particular hyperedges. The global hypergraph objects V and H may be represented by payload VA, VR, and HA, HR in the tombstone implementation, respectively.

The *changeHyperedge* operation takes an existing hyperedge $he(U)$, and the atom sets S^+, S^- that are to be added to and removed from the set U. For simplicity, assume that $S^+ \cap S^- = \emptyset$, $S^+ \cap\ U = \emptyset$ and $S^- \subseteq U$. For readability, we use the set operations of intersection and subset. These conditions can be expressed in terms of the query operations. Note that in the precondition of *changeHyperedge*, we need to check that the set S^+ being added should exist in the (global) hypergraph, whereas the set being deleted S^- should already be in the *mutable* set of the given hyperedge. Note, in the **effect** phase, the atoms from the various sets are removed/added to the set of the hypergraph. Observe that the

 ISBN: 978-1-61208-969-0

atoms are only removed from the set of the hyperedge, but *not* from the (global) hypergraph because hyperedges are formed using references of existing other atoms.

V. Proof of Correctness

Most of the arguments related to 2P-Sets and 2P2P-Graphs [15] carry over in the proof that this specification implements a CRDT. It is easy to show that *add* operations or *remove* operations on unrelated atoms naturally commute. If, however, an atom appears (recursively) within the set of another atom, then adding the second atom must causally follow the addition of the first atom. The delivery order ensures it. The reverse holds for remove operations. Concurrent $add(u)$ and $remove(u)$ operations on the same atom u, and concurrent $remove(u)$ and $add(w)$ [or $add(u)$ and $remove(w)$] operations where there is a some relationship between u, w, are dealt with using the 2P-Set conditions [15], the conditions on adding or removing atoms, and transitivity.

Operations other than the remove operations are independent of the *changeHyperedge*. The tombstone set will ensure that removal prevails over modifications. Modifications to different hyperedges commute. Consider two concurrent modifications to the same hyperedge with changes S_1^+, S_1^- and S_2^+, S_2^- respectively. We claim that the operations can safely commute (refer to the Lemma 1), resulting in set $(U \cup S_1^+ \cup S_2^+) \setminus (S_1^- \cup S_2^-)$. Atoms appearing in the corresponding add set (or removal set) pose no problem. The assumptions about the sets of atoms being added or removed from a given set within each operation allow the commutation.

Lemma 1. *Concurrent* $changeHyperedge(he,\ S_1^+,\ S_1^-)$ *and* $changeHyperedge(he,\ S_2^+,\ S_2^-)$ *commute.*

Proof. According to the $changeHyperedge$ operation, a set of atoms S^+ are added to, and a set of atoms S^- are removed from a hyperedge. Therefore:

$$changeHyperedge\ (he,\ S_1^+,\ S_1^-) = U \cup S_1^+ \text{ and } U \setminus S_1^- = (U \cup S_1^+) \setminus S_1^-$$

Similarly,

$$changeHyperedge\ (he,\ S_2^+,\ S_2^-) = U \cup S_2^+ \text{ and } U \setminus S_2^- = (U \cup S_2^+) \setminus S_2^-$$

The concurrent execution of both the change operations on each replica on the same hyperedge results:

$$changeHyperedge\ (he,\ S_1^+,\ S_1^-)\ ||\ changeHyperedge\ (he,\ S_2^+,\ S_2^-) = (U \cup S_1^+ \cup S_2^+) \setminus (S_1^- \cup S_2^-)\ ||\ (U \cup S_2^+ \cup S_1^+) \setminus (S_2^- \cup S_1^-)$$

Further, the commutative *set-union* operation makes the results equivalent:

$$(U \cup S_1^+ \cup S_2^+) \setminus (S_1^- \cup S_2^-) \equiv (U \cup S_2^+ \cup S_1^+) \setminus (S_2^- \cup S_1^-)$$

Therefore, modification of concurrent operations to the same hyperedge commute. □

VI. Discussion

Our proposed HgCRDT framework has been implemented in our hypergraph-oriented database system. The system works in the realm of an underpinning object-oriented framework, supporting object re-usability, complex objects, data abstraction, encapsulation, and typing-like features. We use *well-defined schema* and *types* to build hypergraphs where higher-order relationships are formulated on top of other existing relationships and entities without violating schematic acyclic dependencies.

Our system ensures the consistency of hypergraph objects among all the replicas of a distributed domain in its *Consistency layer*, after which each replica immediately stores the objects in its local database. Currently, the distribution process works in a multi-threaded environment (#6 threads). The system stores and retrieves hypergraph objects from its storage and retrieval layers that are built atop *HyperGraphDB* [34]. HyperGraphDB is a general-purpose, portable, extensible, and typed data storage mechanism. We use HyperGraphDB to exploit object-level sharing in higher-order and n-ary relationships.

VII. Conclusions

We proposed hypergraphs as a natural candidate structure for representing semi-structured, hierarchical, navigational, complex, higher-order relationships in distributed computing settings. We introduced and specified a new CRDT, a well-formed higher-order recursively-defined mutable hypergraph named HgCRDT, where hypergraphs were modeled using user-defined schema and system-defined object-oriented types. In HgCRDT, the hyperedges themselves were mutable. The HgCRDT is an operation-based specification of 2P2P sets, which works with tombstone sets.

An extension of our approach introduces and implements partial replication in the HgCRDT in a hypergraph-oriented database model built atop HyperGraphDB. However, we have omitted the specification and details pertaining to the partial replication for clarity of exposition. We are in the process of formalizing our approach to prove it algebraically, giving a detailed mathematical proof of our approach; and studying the performance of the replicated hypergraphs, particularly the scalability of the approach, and evaluating the time and space complexity when dealing with a variety of large hypergraphs on real data. We also intend to compare our approach with other possible Hypergraph specifications such as state-based, and other Set CRDTs-based variations, e.g., OR Set.

Acknowledgments: I wish to acknowledge fruitful discussions and collaborated work of Sanjiva Prasad, IIT Delhi.

References

[1] R. Fagin, "Degrees of acyclicity for hypergraphs and relational database schemes," *Journal of the ACM (JACM)*, vol. 30, no. 3, pp. 514–550, 1983.

[2] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis, "On the desirability of acyclic database schemes," *Journal of the ACM (JACM)*, vol. 30, no. 3, pp. 479–513, 1983.

 ISBN: 978-1-61208-969-0

[3] M. Yannakakis, "Algorithms for acyclic database schemes," in *Proceedings of the Seventh International Conference on Very Large Data Bases - Volume 7*, ser. VLDB '81. VLDB Endowment, 1981, p. 82–94.

[4] G. Gallo, G. Longo, S. Pallottino, and S. Nguyen, "Directed hypergraphs and applications," *Discrete applied mathematics*, vol. 42, no. 2-3, pp. 177–201, 1993.

[5] S. Prasad, "Designing for scalability and trustworthiness in mhealth systems," in *International Conference on Distributed Computing and Internet Technology*. Springer, 2015, pp. 114–133.

[6] M. M. Wolf, A. M. Klinvex, and D. M. Dunlavy, "Advantages to modeling relational data using hypergraphs versus graphs," in *2016 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2016, pp. 1–7.

[7] M. Levene and A. Poulovassilis, "An object-oriented data model formalised through hypergraphs," *Data & Knowledge Engineering*, vol. 6, no. 3, pp. 205–224, 1991.

[8] R. Angles and C. Gutierrez, "Survey of graph database models," *ACM Computing Surveys (CSUR)*, vol. 40, no. 1, p. 1, 2008.

[9] S. Klamt, U.-U. Haus, and F. Theis, "Hypergraphs and cellular networks," *PLoS computational biology*, vol. 5, no. 5, pp. 243 – 255, 2009.

[10] E. Ramadan, A. Tarafdar, and A. Pothen, "A hypergraph model for the yeast protein complex network," in *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.*, 2004, pp. 189–.

[11] S. Gilbert and N. Lynch, "Perspectives on the cap theorem," *Computer*, vol. 45, no. 2, pp. 30–36, 2012.

[12] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process." Massachusetts Inst of Tech Cambridge lab for Computer Science, Tech. Rep., 1982.

[13] A.-M. Kermarrec, A. Rowstron, M. Shapiro, and P. Druschel, "The icecube approach to the reconciliation of divergent replicas," in *Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*, 2001, pp. 210–218.

[14] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski, "A comprehensive study of convergent and commutative replicated data types," INRIA Centre Paris-Rocquencourt, Research Report RR-7506, 2011.

[15] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski, "Conflict-free replicated data types," in *Stabilization, Safety, and Security of Distributed Systems*, X. Défago, F. Petit, and V. Villain, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 386–400.

[16] V. B. Gomes, M. Kleppmann, D. P. Mulligan, and A. R. Beresford, "Verifying strong eventual consistency in distributed systems," *Proceedings of the ACM on Programming Languages*, vol. 1, no. OOPSLA, p. 109, 2017.

[17] R. Brown, Z. Lakhani, and P. Place, "Big(ger) Sets: decomposed delta CRDT Sets in Riak," in *Proceedings of the 2nd Workshop on the Principles and Practice of Consistency for Distributed Data*. ACM, 2016, p. 5.

[18] S. Weiss, P. Urso, and P. Molli, "Logoot-undo: Distributed collaborative editing system on p2p networks," *IEEE transactions on parallel and distributed systems*, vol. 21, no. 8, pp. 1162–1174, 2010.

[19] N. Preguiça, J. M. Marquès, M. Shapiro, and M. Letia, "A commutative replicated data type for cooperative editing," in *2009 29th IEEE International Conference on Distributed Computing Systems*. IEEE, 2009, pp. 395–403.

[20] M. Levene and A. Poulovassilis, "The hypernode model and its associated query language," in *Proceedings of the 5th Jerusalem Conference on Information Technology, 1990.'Next Decade in Information Technology'*. IEEE, 1990, pp. 520–530.

[21] M. Owrang and L. L. Miller, "An approach for integration of data processing in a distributed enviornment," in *Proceedings of the 17th conference on ACM Annual Computer Science Conference*, 1989, pp. 358–367.

[22] F. Laux, "The typed graph model," in *The Twelfth International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA)*, pp. 13 – 19. [Online]. Available: https://www.iaria.org/conferences2020/DBKDA20.html

[23] W. Jiang, J. Qi, J. X. Yu, J. Huang, and R. Zhang, "Hyperx: A scalable hypergraph framework," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 909–922, 2018.

[24] B. Shao, H. Wang, and Y. Li, "Trinity: A distributed graph engine on a memory cloud," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. ACM, 2013, pp. 505–516.

[25] R. Brown, S. Cribbs, C. Meiklejohn, and S. Elliott, "Riak dt map: A composable, convergent replicated dictionary," ser. PaPEC. ACM, 2014.

[26] M. Kleppmann and A. R. Beresford, "A Conflict-Free Replicated JSON Datatype," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 10, pp. 2733–2746, 2017.

[27] A. Leijnse, P. S. Almeida, and C. Baquero, "Higher-Order Patterns in Replicated Data Types," in *Proceedings of the 6th Workshop on Principles and Practice of Consistency for Distributed Data*. ACM, 2019.

[28] A. Hall, G. Nelson, M. Thiesen, and N. Woods, "The causal graph crdt for complex document structure," in *Proceedings of the ACM Symposium on Document Engineering*, 2018.

[29] P. S. Almeida, A. Shoker, and C. Baquero, "Efficient State-based CRDTs by Delta-mutation," in *International Conference on Networked Systems*. Springer, 2015, pp. 62–76.

[30] P. Sérgio Almeida, A. Shoker, and C. Baquero, "Delta state replicated data types," *Journal of Parallel and Distributed Computing*, vol. 111, pp. 162–173, 2018.

[31] L. D. Ibáñez, H. Skaf-Molli, P. Molli, and O. Corby, "Synchronizing semantic stores with commutative replicated data types," in *Proceedings of the 21st International Conference on World Wide Web*. ACM, 2012.

[32] L.-D. Ibáñez, H. Skaf-Molli, P. Molli, and O. Corby, "Live linked data: synchronising semantic stores with commutative replicated data types," *International Journal of Metadata, Semantics and Ontologies 4, 8*, vol. 8, no. 2, pp. 119–133, 2013.

[33] "Acl2020: General conference statistics," https://acl2020.org/blog/general-conference-statistics/, 2008, [Online; accessed on 06-April-2022].

[34] B. Iordanov, "Hypergraphdb: a generalized graph database," in *International conference on web-age information management*, ser. WAIM'10. Springer, 2010, pp. 25–36.

 ISBN: 978-1-61208-969-0

Comparison of Experiment Tracking Frameworks in Machine Learning Environments

Tim Budras[†], Maximilian Blanck[*], Tilman Berger[*], and Andreas Schmidt[†‡],
[*] inovex GmbH, Karlsruhe
Email: {mblanck, tberger}@inovex.de
[†] Department of Computer Science and Business Information Systems,
Karlsruhe University of Applied Sciences
Karlsruhe, Germany
Email: {buti1021, andreas.schmidt}@h-ka.de
[‡] Institute for Automation and Applied Computer Science
Karlsruhe Institute of Technology
Karlsruhe, Germany
Email: andreas.schmidt@kit.edu

***Abstract*—The machine learning market is growing and machine learning is increasingly being used productively. Because of this, more and more tools have been developed in the past with the aim of supporting machine learning in practice. One type of these tools is called *experiment tracking tools*. Their objective is to keep track of the information generated by different experiment runs so that the information can be used later, for example, to find the best experiment run. Within the context of a bachelor thesis, a pre-selection of 20 systems was made and then 4 of them were selected for a more in-depth analysis and their characteristics were examined in more detail. This paper summarizes the most important findings of this thesis.**

***Index Terms*—Machine Learning; Experiment Tracking; Development Environment**

I. Introduction

The machine learning market is growing strongly. According to MarketsandMarkets [1], it is "expected to grow from USD 1.03 billion in 2016 to USD 8.81 billion by 2022". As a result of this growth, tools have been developed in recent years to help develop machine learning models and put them into production. However, due to the fact that the use of machine learning in productive software is relatively new, tools and conventions are less settled and less commonly applied than in traditional software development. Warden [2] uses the term "machine-learning-reproducibility-crisis" to describe that the tools to meet these needs are often not applied in practice.

With regard to tracking data, parameters, models and results, numerous products with different focuses and strengths have been developed. Tools that focus on saving information around the model training and development process are often referred to as *experiment tracking tools*. But as stated in a Kaggle survey [3], in a large amount of scenarios these relatively new tools remain unused and tracking is either done manually or not done at all.

The rest of the paper is structured as follows: In Section II we explain the *machine learning lifecycle* and what artifacts needs to be tracked in the context of an experiment. Based on these findings we present the general architecture for experiment tracking tools and formulate the most important requirements in Section III. In Section IV a number of concrete tools are presented and compared. The papers is finished with a Conclusion in Section V.

II. Background

In this section, a set of basic insights required for understanding tracking tools in the field of machine learning will be presented.

A. The Machine Learning Lifecycle

The different phases and steps around the productive use of a machine learning model have been described by different authors using different terms. One of these terms is the *machine learning lifecycle*. Garcia et al. [4] describe the machine learning lifecycle as a three-phase process as shown in Figure 1. The first phase is the *pipeline development*. During this iterative phase, the data preprocessing, exploration and visualization is done, model designs are chosen and models get trained with different configurations and hyperparameters. The authors emphasize that the model is not the important product of the first phase, but the pipeline, which can be reused to create a model from a data set. This pipeline can be used later in the second phase *training* (middle), to train and validate the model used for inference. The last phase (right) is called *inference*. Here, the prediction service (which includes the data preprocessing as well as the model used for inference) returns a prediction for a given user input. The prediction service provides information on the made predictions, which can be used for later trainings. The authors mention that the different stages are often managed by different teams.

B. Experiment Tracking

Langley [5] describes machine learning as an experimental science and compares the process of finding a good model to the empirical sciences of physics and chemistry. This aligns with the results from interviews Hill et al. [6] conducted

 ISBN: 978-1-61208-969-0

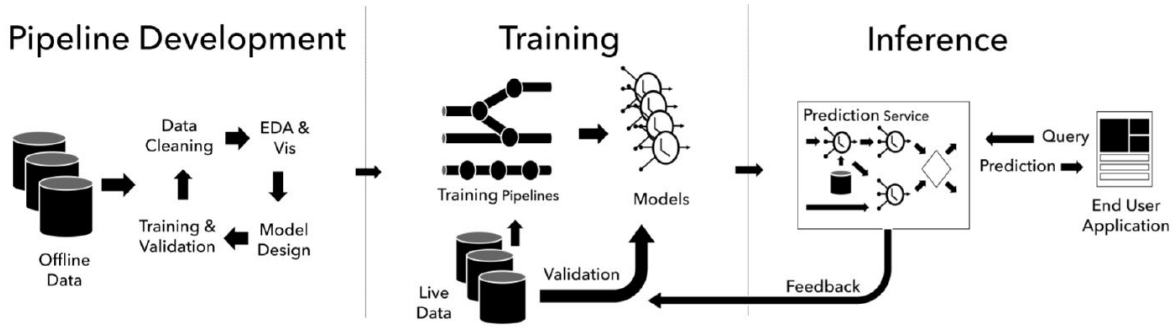


Fig. 1. Machine Learning Lifecycle (from [4])

with various machine learning practitioners in 2016. Seven out of seven interviewees experienced the need ”to resort to basic trial and error”. Langley defines an experiment as the process of examining the effect of varying one or more independent variables on some dependent variables [5]. Hence, an experiment consists of multiple runs. According to Vartak et al. [7], ”data scientist often built hundreds of models before arriving at one that met some acceptance criteria”. Each model built can be seen as the dependent variable of a run. However, experiment tracking tools can also be used in the pipeline development phase, introduced by Garcia et al. [4], which does not produce a model, but a training pipeline. In this case, the dependent variable would be the training pipeline. Therefore, the following definition of a(n experiment) run is used in this paper:

Definition (Experiment): A run is a part of an experiment, it has a specific set of independent variables that produces a model or a training pipeline. An experiment is a collection of runs that try to solve the same problem or business task. The objective of an experiment is to find the set of independent variables resulting in the best dependent variable(s).

It should be noted that usually in practice it is not possible or at least not economically feasible to find the best independent variables [8].

Various possibilities exist to assess the quality of a model. A common possibility is to calculate a metric for prediction quality (such as accuracy) on a data set not used for training. However, additional (nonfunctional) quality measures might exist, e.g. the inference time, the training time or the explainability of a prediction.

Due to the fact that the number of experiment runs might be enormous, it is very helpful to track the experiment and its runs. The term *experiment tracking* describes the process of saving the information related to the experiment and its runs, to allow further evaluation. Although typically the verb *to track* is used in combination with experiments, some tools evaluated in this work have functionalities that use the words *log* or *logger*. Thus, both terms are treated as synonyms in this work. In its easiest version, tracking can be done manually, alternatively one of the tools presented in Section IV can be used. Either way, tracking experiments brings multiple advantages:

Keeping track of all the runs makes it easy to find the best variables. Additionally, it is easy to see which sets of independent variables have already been tried out or might be worth trying out in the future. This is especially helpful if the work is done in teams, or if the responsible person changes. With the right tool, tracked experiments can be easily compared. If a model is used in production, it can be very helpful to have the information available on how the model was created. Another advantage – which applies especially to research – is the fact, that results may need to be reproduced. Furthermore, establishing the use of an experiment tracking tool in a company or a project provides the benefit of a structured way to access the data generated during experimentation, regardless of the individuals responsible for the experiments.

C. Reproducibility Requirements

In a reproducibility challenge, Pineau showed that most challenge attendees found it at least reasonably difficult to reproduce the result of a paper of the *International Conference on Learning Representations 2018* [9]. Pineau also published a machine learning reproducibility checklist [10], which is supposed to help increase the reproducibility of experiments. Tatman et al. [11] define three levels of reproducibility for research: low, medium and high reproducibility. The lowest level of reproducibility is achieved by publishing the paper. According to the authors, the medium level is achieved, when the code is published along with the used data. The highest level can be reached by additionally providing the environment.

In the following subsections the requirements for reproducibility introduced by Tatman et al. [11] as well as the terms *hyperparameters* and *metrics* will be explained in detail.

1) Code: Similar to traditional programming, machine learning highly depends on the source code. There are several tools to effectively version source code. A developer survey by StackOverflow in 2018 [12] showed that almost 90 % of the developers use Git as a version control system. There is no valid reason to not track the code used in machine learning projects with Git. However, in a fast developing process, experiment runs might be executed, without committing the code beforehand. This would lead to a lack of reproducibility, as Git needs a commit to restore a state of the code.

 ISBN: 978-1-61208-969-0

2) Data: Besides the code, data plays an essential role in machine learning, because different data can lead to different results. As the kind of data depends on the business task, the data format varies. Common data formats are text, image or video. Due to the partly large data resources, a suitable tool for the efficient storage of different variants of a data resource should be used.

3) Environment: Providing information about the environment is certainly only necessary for some use cases. However, it can contain important information of the original run, such as the used hardware, the used operating system or the software dependencies. Thus, keeping track of the environment can be helpful to reproduce a run. Tatman et al. [11] propose three possibilities to share the environment: Either by using a hosted service, or by providing a container or virtual machine, which includes all dependencies. At minimum, the used libraries and their versions should be tracked.

4) Hyperparameters: According to Bergstra et al. [13], hyperparameters configure the machine learning algorithm before training, whereas, in the present paper, any kind of configuration parameters of the experiment run (not only the machine learning algorithm) will be considered as hyperparameters. As any change in configuration might result in different results, it is recommended to track as many hyperparameters as possible. Although hyperparameters are often tracked implicitly when they are defined in the code and the code is versioned, hyperparameters should be tracked explicitly to allow easier comparison.

5) Metrics: A metric is an evaluation measure calculated to quantify "the effectiveness of a complete application that includes machine learning components" [8]. Most of the times, metrics will be calculated based on a model's predictions on data that has not been used for training. Different metrics with varying strengths and weaknesses exist. For classification tasks for example, accuracy or precision can be used. Accuracy is defined as the fraction of correct predictions out of all predictions [8]. Metrics can be used to compare different runs of an experiment and can be considered as one of the dependent variables of the experiment. Which type of metric is used, is not important regarding experiment tracking.

III. Experiment Tracking Tools

The main goal of experiment tracking is to save information during experimentation in order to be able to access it later. As a result, most experiment tracking tools consist of at least three components, as shown in Figure 2. Some kind of client software – for example a Python library – is required to store the tracked information during experimenting on a persistent data storage or send it to a server. The data can often be retrieved programmatically through the client or be viewed in a Graphical User Interface (GUI). The exact functionality of those components differs between the available tools.

A. Requirements

As already discussed in Subsection II-B, tracking of code, data, the used environment, hyperparameters, and metrics are elementary requirements for such a tool. Additional requirements examined in our research also include the following aspects:

1) Storing of Models: Training a model can take a long time. Therefore, the models should be stored and linked to the hyperparameters and metrics. This avoids time consuming retraining e.g., if a model should be evaluated on new data.

2) Accessibility of Tracked Information: Tracking is a prerequisite, however the tracked data will only provide value, if the tracked information can be accessed in a simple yet powerful way. This includes a user interface which provides a clear and customizable overview of all runs, as well as the possibility to compare runs in depth. Filtering the runs with easy but rich querying options is also part of this requirement. Besides that, the tool should provide a possibility to create and show plots. If additional interfaces, e.g., an API, exist, they will be useful as well.

3) Collaboration: According to Tabladillo et al. [14], bringing data science projects to production requires different tasks. For this reason, data science projects are often worked on in teams composed of different roles. Therefore, the tool should facilitate collaborative work. This includes the possibility of viewing existing results of different team members and adding new results by executing new runs. To achieve this, a form of access management is required.

4) Initial Setup and Infrastructure: Because tracking machine learning experiments should facilitate the work of the teams, tools will only be taken into consideration if they have low barriers to entry. Thus, this requirement describes the initial investment needed to set up and use the tool. The initial setup is everything that does not need to be repeated if the same tool is used in another project (given the projects can use the same infrastructure). As cloud tools might have an advantage concerning the initial setup, it must be kept in mind, that saving data off-premises might not be a possibility due to legal or corporate regulations.

5) Ease of Integration: Similar to the previous requirement this requirement concerns user-friendliness. Yet, unlike the initial setup and infrastructure, the ease of integration describes how easy it is to include the tool into a specific project. This means, for example, project-specific configuration or source code changes.

IV. Examined Tools

In a market research, the following tools with experiment tracking functionality were identified.

- Aim [15]
- Amazon SageMaker Experiments [16]
- Azure Machine Learning [17]
- ClearML [18]
- Comet [19]
- DAGsHub [20]
- DominoDataLab [21]
- DVC Studio [22]
- Guild AI [23]
- H2O MLOps [24]

 ISBN: 978-1-61208-969-0

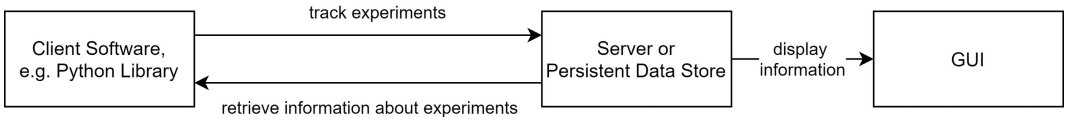


Fig. 2. General Architecture of an Experiment-Tracking-Tool

- MLflow [25]
- Neptune [26]
- Paperspace Gradient [27]
- Polyaxon [28]
- Sacred [28] in combination with Omniboard, Incense or Sacredboard (GUIs)
- TensorBoard [29]
- Valohai [30]
- Verta [31]
- Vertex AI [32]
- Weights & Biases [33]

The research was conducted online, using search engines, blogs, forums, as well as the websites of the respective tools.

To allow an in-depth evaluation of the tools in the scope of this work, the tools listed previously have to be limited to a reasonable amount. The tools were selected in consultation with a project team at inovex, actually developing a multilingual and multidomain Conversational AI. The selection was influenced by requirements given from the project team. In this process, MLflow, ClearML, Neptune and DAGsHub were adopted for a more detailed evaluation. MLflow was selected because it is one of the most established and widely used tools. ClearML was assessed because of its wide range of operating options. It can be used for free (even in small teams) as a hosted option, operated self-hosted for free, but also be used with a paid plan. The most important argument for choosing Neptune was that it promises an effortless setup. The last option evaluated was DAGsHub, as it makes use of Data Version Control (DVC) [34] for versioning data, like the project. In the next subsections each tool will be evaluated based on the requirements defined in Subsection III-A and an exemplary integration will be provided.

A. MLflow

The open-source tool MLflow is developed by Databricks.

```
1 import mlflow
2 mlflow.set_tracking_uri("postgresql://postgres:
      postgres@172.3...")
3 mlflow.set_experiment("MyProject") #group runs
4 with mlflow.start_run() as run:
5     hyperparams = {"lr": 0.01,}
6     mlflow.log_params(hyperparams)
7     #Training placeholder, model stored in var model
8     mlflow.pytorch.log_model(model, "log_r",)
9     mlflow.log_metric("acc", 0.99)
```

Listing 1. MLflow example code

To start tracking with MLflow, a run has to be started as shown in Listing 1. By using a context manager, the run will be ended automatically (line 4). MLflow differentiates between metrics and params; both can be logged to MLflow by using the respective function. MLflow provides functions to log one value (line 9), or to log multiple values (here, a dictionary is passed, as the only parameter and the name and values of the dictionary will be used (line 6). Grouping multiple runs together allows easy viewing and comparison in the GUI. This can be achieved by setting up an experiment (line 3). Metadata (params, metrics, etc.) are by default stored in a local text file. However, other possibilities exist; such as saving them in a SQL Database, which can be achieved by specifying a tracking URI (line 2). By default, models logged with MLflow are stored in the local file system. However, it is possible to change the location, e.g., to an S3 bucket.

The MLflow GUI in Figure 3 shows all the hyperparameters and metrics in a clear table. Runs of the same experiment can be compared and metrics are automatically plotted. In addition to the GUI, data tracked with MLflow can be retrieved via Python, R, Java and REST APIs. MLflow does not provide a dedicated way to keep track of the data used for training. It does not support automated tracking of the environment either. It can be used for free in teams, however, this requires shared data storage, which has to be set up by yourself.

B. Neptune

Neptune is a tool developed by Neptune Labs. While the Client Software (Python package) is open-source, the server code is not publicly available. Free as well as paid plans exist. To get started with Neptune, an account has to be created at neptune.ai and an API token has to be generated. To track experiments, a project (similar to an experiment in MLflow) has to be created in the Neptune Web App. After those setup steps, Neptune is ready for use.

```
1 import neptune.new as neptune
2 run = neptune.init(project="tbud/MyProject")
3 hyperparams = {"lr": 0.01,}
4 run["hyperparams"] = hyperparams
5     #Trainingloop placeholder
6     run["loss/train"].log(the_current_loss)
7 torch.save(model, "log_r.mdl")
8 run["model"].upload("log_r.mdl")
9 run["acc"] = 0.99
```

Listing 2. Neptune example code

Listing 2 shows the integration of Neptune, after importing the new Neptune API, we can initialize a run and assign it to a project (line 2). Neptune does not differentiate between metrics and hyperparameters. To log values with Neptune, a notation with square brackets and strings as keys (e.g., run[”some_key”]) is used, which is similar to adding new values to a dict (line 9). To track series such as the loss, the log function has to be used (line 6). This automatically generates a plot in the GUI. To upload a trained model, it first has to

 ISBN: 978-1-61208-969-0

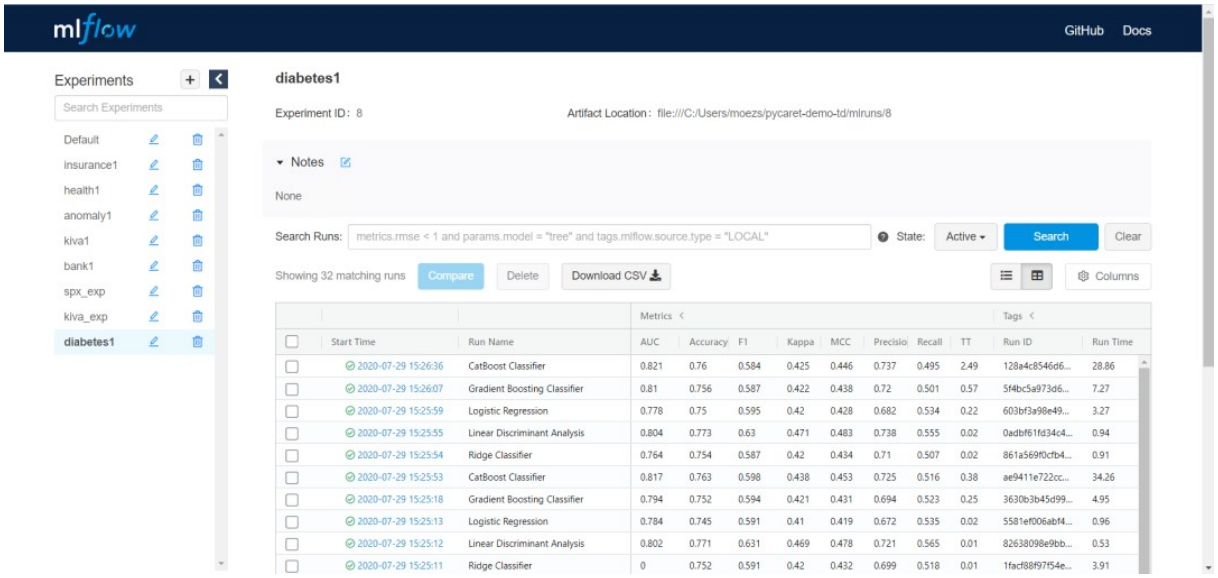


Fig. 3. MLflow GUI (from [35])

be saved locally and can then be uploaded to Neptune using the upload (line 7).

The GUI of Neptune (Figure 4) looks similar to the MLflow GUI. It includes all the basic functionalities that MLflow has, but also has additional nice-to-have features, such as query completion for filtering or an option to save customized views. The data can also be retrieved through the Python API. Similar to MLflow, Neptune's focus is tracking metrics and hyperparameters. The setup is easier as with MLflow, however, using Neptune raises Data Governance questions, because data is stored on Neptune servers, outside your own company. For single users Neptune can be used for free. When working in teams the prize is calculated based on the usage.

C. ClearML

ClearML is an open-source tool developed by Allegro AI, it was formerly known as Allegro Trains. Multiple options to operate ClearML exist, it can be self-managed for free, used with a free as-a-Service plan for up to three team members or used with a paid plan.

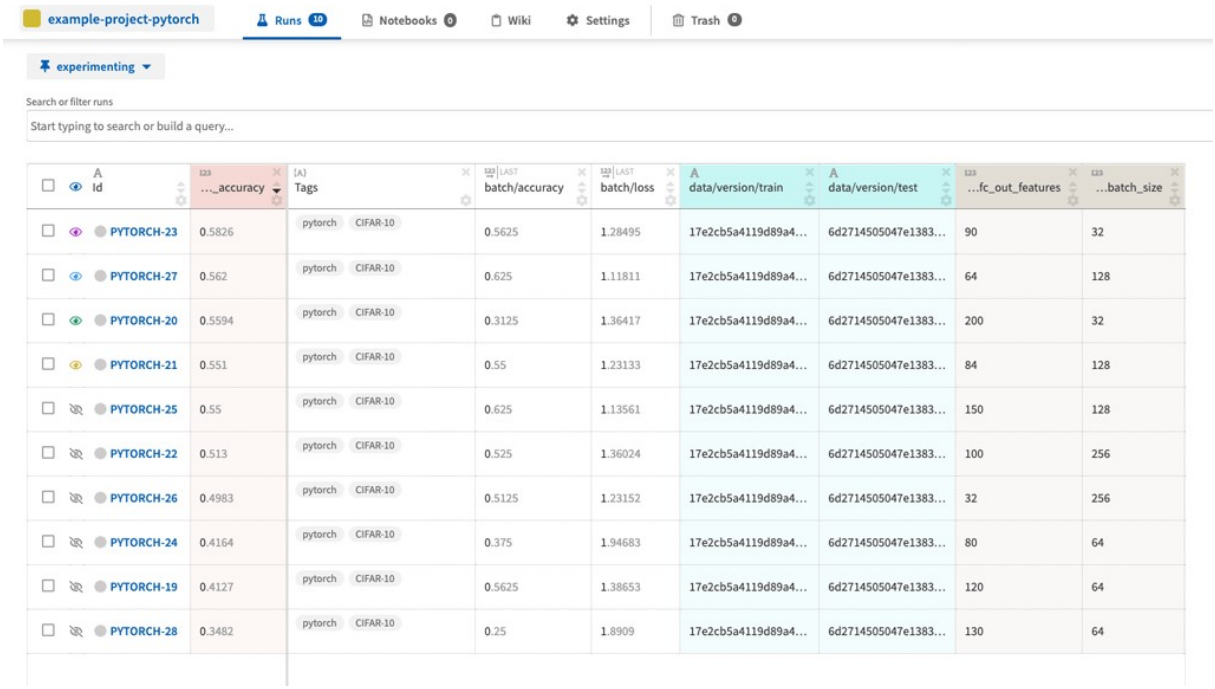
```
1 from clearml import Task, Logger, Dataset
2 path = Dataset.get(dataset_project="MyProject/data",
      dataset_name="ds_1").get_local_copy()
3 task = Task.init(project_name="MyProject", task_name
     ="Task1", reuse_last_task_id=False, output_uri="
     gs://MyProject",)
4 hyperparams = {"lr": 0.01,}
5 task.connect(hyperparams)
6 #Training placeholder, model stored in var model
7 torch.save(model, "log_r.mdl")
8 task.get_logger().report_scalar("model", "accuracy",
      0.99, 0)
```

Listing 3. ClearML example code

Besides its hyperparameter and metric tracking capabilities, ClearML provides a possibility to efficiently store and manage large datasets. It works similar to DVC [34]. This allows versioning datasets even for binary files. A simple example of the integration into code is given in Listing 3. To get the local path to a dataset managed with ClearML, the dataset has to be queried with the `Dataset.get()` function (line 2). The `get_local_copy()` (line 2) function ensures that a local copy is available and returns the path, which can then be used for training. In ClearML, a task is similar to a run in MLflow and describes something that is executed and should be tracked. In line 3, a task is initialized and assigned to a project. Setting `reuse_last_task_id` to False ensures that this task will not override an old task. The `output_uri` specifies the location for the artifacts (e.g., the model) and is in this example set to a Google Cloud Storage. By initializing a task, the tracking is automatically started. ClearML allows logging hyperparameters by connecting an object to a task (line 5). When a model is saved locally, ClearML automatically uploads it to the artifact store and connects it to the task (line 7). Metrics can be reported to a logger, where the first argument is the title of the plot, the second is the name of the series, the third is the value and the last is the iteration (x-coordinate). It should be noted that executing `Task.init` automatically tracks the used python packages and their versions, providing an additional amount of information.

Figure 5 shows a screenshot of the GUI. While the overview table of the experiments looks similar to Neptune and MLflow, the detailed view of the task is very nested and can overwhelm new users. This is in our opinion the biggest downside of ClearML compared to the other tools: due to its huge amount of possibilities, it requires more time to familiarize. However, we think this time is well invested since ClearML provides a lot of options and possibilities for the user. Besides the Python API data collected with ClearML can also be retrieved with a REST API.

 ISBN: 978-1-61208-969-0



Id	..._accuracy	Tags	batch/accuracy	batch/loss	data/version/train	data/version/test	...fc_out_features	...batch_size
PYTORCH-23	0.5826	pytorch CIFAR-10	0.5625	1.28495	17e2cb5a4119d89a4...	6d2714505047e1383...	90	32
PYTORCH-27	0.562	pytorch CIFAR-10	0.625	1.11811	17e2cb5a4119d89a4...	6d2714505047e1383...	64	128
PYTORCH-20	0.5594	pytorch CIFAR-10	0.3125	1.36417	17e2cb5a4119d89a4...	6d2714505047e1383...	200	32
PYTORCH-21	0.551	pytorch CIFAR-10	0.55	1.23133	17e2cb5a4119d89a4...	6d2714505047e1383...	84	128
PYTORCH-25	0.55	pytorch CIFAR-10	0.625	1.13561	17e2cb5a4119d89a4...	6d2714505047e1383...	150	128
PYTORCH-22	0.513	pytorch CIFAR-10	0.525	1.36024	17e2cb5a4119d89a4...	6d2714505047e1383...	100	256
PYTORCH-26	0.4983	pytorch CIFAR-10	0.5125	1.23152	17e2cb5a4119d89a4...	6d2714505047e1383...	32	256
PYTORCH-24	0.4164	pytorch CIFAR-10	0.375	1.94683	17e2cb5a4119d89a4...	6d2714505047e1383...	80	64
PYTORCH-19	0.4127	pytorch CIFAR-10	0.5625	1.38653	17e2cb5a4119d89a4...	6d2714505047e1383...	120	64
PYTORCH-28	0.3482	pytorch CIFAR-10	0.25	1.8909	17e2cb5a4119d89a4...	6d2714505047e1383...	130	64

Fig. 4. Neptune GUI (from [26])

D. DAGsHub

In contrast to the other presented tools, DAGsHub offers a different approach. It makes use of existing open-source technologies and provides unified storage and GUI for them (however, DAGsHub itself is not open-source):

- DVC [34] is used to keep track of the data and models.
- Git keeps track of the code.
- MLflow or the DAGsHub Client can be used to track hyperparameters and metrics.

The interaction of the different tools is presented in Figure 6.

Beside DAGSHub's own client, MLFlow can be used to track hyperparameters and metrics. In this case the integration into code looks like in Listing 1. The most important advantages of DAGsHub are the unified storage and the efficient handling of variants of datasets using DVC.

The GUI of DAGsHub in Figure 7 is familiar to GitHub users, but additionally includes a data section, as well as an overview of the experiment runs as known from MLflow. With a free DAGsHub plan, the number of collaborators and storage is limited. Paid plans exist, which allow working in bigger teams. DAGsHub probably has the most potential for teams that already use DVC and/or MLflow and want to keep using the tools but would benefit from unified storage and GUI.

E. Comparison

Table IV-E shows a comparison for most of the defined requirements. As tracking the code is done with Git most of the times and tracking the hyperparameters and metrics and the ease of integration are on a similar level for all four tools, these defined requirements are not included in the table. The tools have different strengths and weaknesses when it comes to ease of use, pricing and more advanced requirements, such as tracking data or computational environment. MLflow has a well-structured API and can be used for free. Neptune, on the other hand, offers a simple setup and highly functional GUI but requires a paid plan when used as a team. In comparison to the two previous tools, ClearML handles the tracking of data and the computational environment, taking care of all requirements. Additionally, it is open-source and can be self-hosted or used as a free or paid Service. DAGsHub can be considered as a good choice for teams already using DVC and MLflow who like to have unified storage and GUI.

V. Conclusion

This paper showed the benefits of tracking machine learning experiments. After presenting 20 tools with functionalities to track experiments which have been identified in a market research. Requirements for machine learning experiment tools were defined based on the needs of an industrial data science project and 4 tools have been evaluated in detail. This evaluation has shown that the right choice of an experiment tracking tool depends on the specific requirements, and identified ClearML as an open source tool that meets most of the requirements.

Due to the quickly changing market of experiment tracking tools, new tools might be released or existing tools might receive new functionality. As a result, further research, also of tools not evaluated in this paper, might be of use.

 ISBN: 978-1-61208-969-0

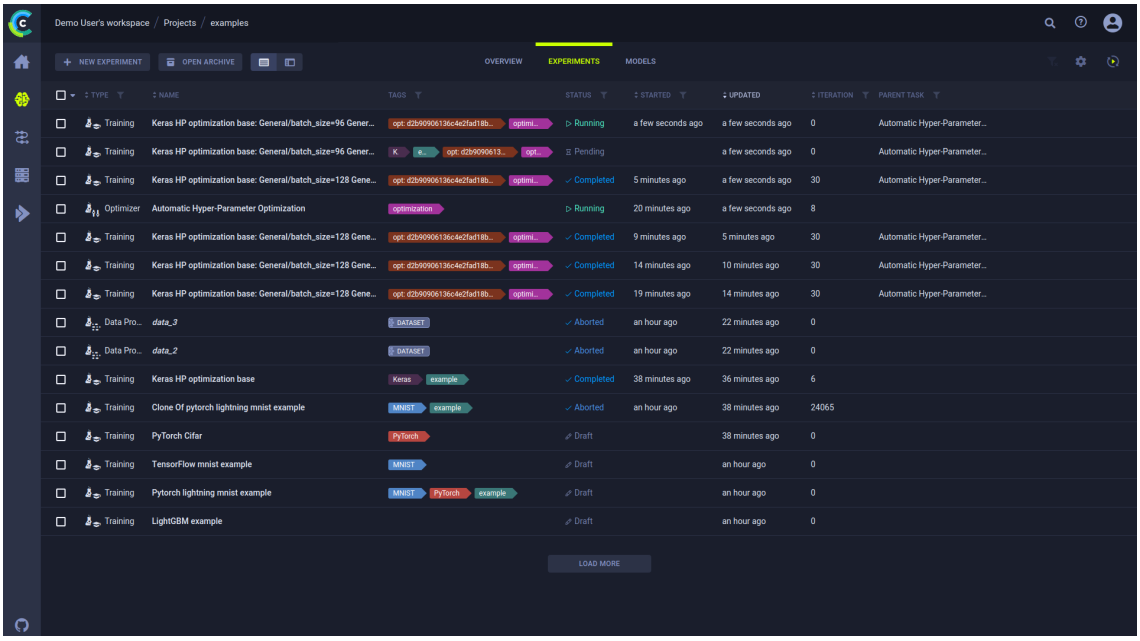


Fig. 5. ClearML GUI (from [36])

TABLE I
COMPARITIVE OVERVIEW OF MLFLOW, NEPTUNE, CLEARML AND DAGSHUB

	MLflow	**Neptune**	**ClearML**	**DAGsHub**
Evaluated version	1.17	0.9.18	1.0	as of June 2021
Data	no dedicated functionality provided	no dedicated functionality provided	Data Managing and Versioning with ClearML Data	Data Managing and Versioning with DVC
Environment	encourages the user to do it manually (MLflow Projects)	no dedicated functionality provided	automatically keeps track of the installed python packages and their versions	no dedicated functionality provided
Storing models	easily possible	model has to be stored locally first and can then be uploaded	automatically uploaded if saved locally	possible to store models with DVC, commit required for every upload
Accessibility of tracked information	basic GUI as well as Python, R, Java and REST APIs	highly customizable & advanced GUI as well as a Python API	advanced GUI as well as a Python API	unified GUI for data, code, and experiments, no Python API for retrieving experiment data
Collaboration	possible, requires a shared data storage	possible with a paid account	possible, user limit depends on the operation mode, unlimited for self-hosting	free for public repositories, not free of charge for private repositories
Initial setup and infrastructure	setting up a database or shared file storage is required for collaborative use	easy setup, as the user does not have to take care of the infrastructure	hosted as well as self-hosting options exist, images to make the setup easier exist	easy setup if DAGsHub is used as Git and DVC storage

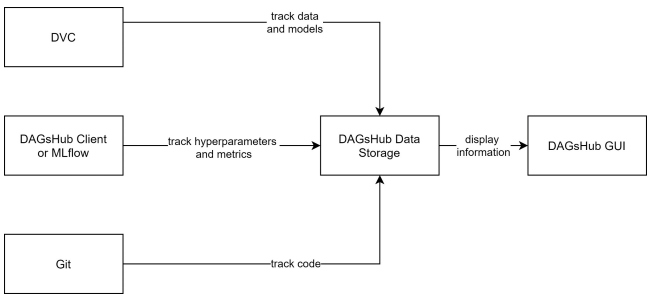


Fig. 6. DAGsHub Architecture

ACKNOWLEDGMENT

The work was carried out in the course of the bachelor thesis [37] of the first author at the company inovex GmbH.

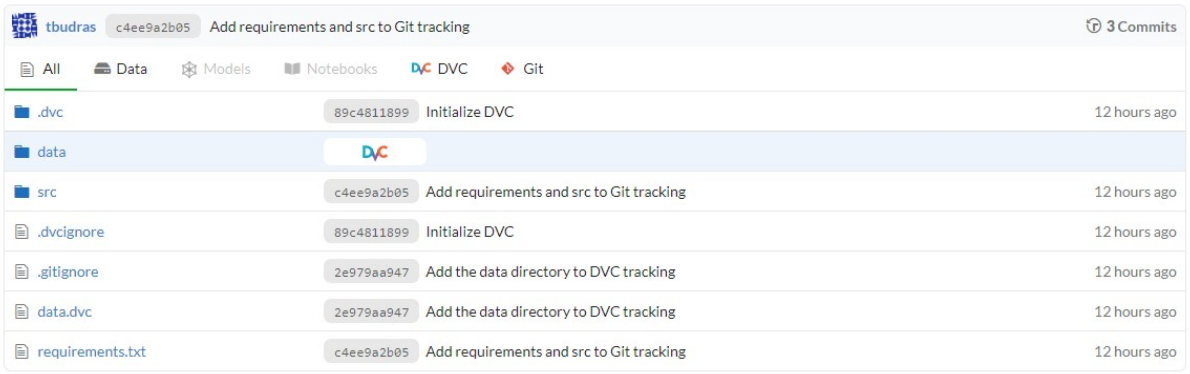


Fig. 7. DAGsHub GUI

REFERENCES

[1] Machine learning market. [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/machine-learning-market-263397704.html (Accessed 2021-07-19).

[2] P. Warden. The machine learning reproducibility crisis. [Online]. Available: https://petewarden.com/2018/03/19/the-machine-learning-reproducibility-crisis/ (Accessed 2021-03-11).

[3] State of data science and machine learning 2020. [Online]. Available: https://www.kaggle.com/kaggle-survey-2020 (Accessed 2021-03-17).

[4] R. Garcia, V. Sreekanti, N. Yadwadkar, D. Crankshaw, J. E. Gonzalez, and J. M. Hellerstein, "Context: The missing piece in the machine learning lifecycle," *KDD CMI Workshop*, vol. 114, pp. 32–38, 2018.

[5] P. Langley, "Machine learning as an experimental science," *Machine Learning*, vol. 3, no. 1, pp. 5–8, 1988. [Online]. Available: https://doi.org/10.1023/A:1022623814640

[6] C. Hill, R. Bellamy, T. Erickson, and M. Burnett, "Trials and tribulations of developers of intelligent systems: A field study," in *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2016, pp. 162–170, ISSN: 1943-6106.

[7] M. Vartak, H. Subramanyam, W.-E. Lee, S. Viswanathan, S. Husnoo, S. Madden, and M. Zaharia, "ModelDB: a system for machine learning model management," in *Proceedings of the Workshop on Human-In-the-Loop Data Analytics - HILDA '16*. ACM Press, 2016, pp. 1–3. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2939502.2939516 (Accessed 2021-03-10).

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[9] J. Pineau, "Reproducibility, reusability, and robustness in deep reinforcement learning," Paper presented at the meeting of ICLR 2018, 2018. [Online]. Available: https://www.youtube.com/watch?v=Vh4H0gOwdIg

[10] J. Pineau, "The machine learning reproducibility checklist," 2020. [Online]. Available: https://www.cs.mcgill.ca/~jpineau/ReproducibilityChecklist.pdf

[11] R. Tatman, J. VanderPlas, and S. Dane, "A practical taxonomy of reproducibility for machine learning research," 2nd Reproducibility in Machine Learning Workshop at ICML 2018, Stockholm, Sweden., 2018.

[12] Stack Overflow, "Stack overflow developer survey results 2018," 2018. [Online]. Available: https://insights.stackoverflow.com/survey/2018/#work-_-version-control

[13] J. Bergstra, D. Yamins, and D. D. Cox, "Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms," Proceedings of the 12th Python in Science Conference in Science Conference (SCIPY 2013).

[14] M. Tabladillo, A. Arora, and C. Gronlund, "What is the Team Data Science Process?" [Online]. Available: https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/overview (Accessed 2021-05-10).

[15] Aim. [Online]. Available: https://aimstack.io (Accessed 2021-07-31).

[16] Amazon sagemaker. [Online]. Available: https://aws.amazon.com/sagemaker/features/ (Accessed 2021-07-31).

[17] Azure machine learning. [Online]. Available: https://docs.microsoft.com/de-de/azure/machine-learning/how-to-track-monitor-analyze-runs?tabs=python (Accessed 2021-07-31).

[18] Clearml. [Online]. Available: https://clear.ml (Accessed 2021-05-11).

[19] Comet. [Online]. Available: https://www.comet.ml/site/ (Accessed 2021-07-31).

[20] Dagshub. [Online]. Available: https://dagshub.com (Accessed 2021-07-31).

[21] Dominodatalab. [Online]. Available: https://www.dominodatalab.com (Accessed 2021-07-31).

[22] Dvc studio. [Online]. Available: https://studio.iterative.ai (Accessed 2021-07-31).

[23] Guild ai. [Online]. Available: https://guild.ai (Accessed 2021-07-31).

[24] H2o mlops. [Online]. Available: https://www.h2o.ai/products/h2o-mlops/ (Accessed 2021-07-31).

[25] Mlflow. [Online]. Available: https://mlflow.org (Accessed 2021-07-31).

[26] Neptune. [Online]. Available: https://neptune.ai/product (Accessed 2022-04-07).

[27] Paperspace gradient. [Online]. Available: https://gradient.paperspace.com (Accessed 2021-07-31).

[28] Polyaxon. [Online]. Available: https://polyaxon.com (Accessed 2021-07-31).

[29] Tensorboard. [Online]. Available: https://www.tensorflow.org/tensorboard/ (Accessed 2021-07-31).

[30] Valohai. [Online]. Available: https://valohai.com (Accessed 2021-07-31).

[31] Verta. [Online]. Available: https://www.verta.ai (Accessed 2021-07-31).

[32] Vertex ai. [Online]. Available: https://cloud.google.com/vertex-ai (Accessed 2021-07-31).

[33] Weights & biases. [Online]. Available: https://wandb.ai/site (Accessed 2021-07-31).

[34] Data version control - documentation. [Online]. Available: https://dvc.org/doc (Accessed 2022-03-12).

[35] Pycaret logging with mlflow. [Online]. Available: https://pycaret.gitbook.io/docs/get-started/functions/initialize#experiment-logging (Accessed 2022-04-06).

[36] Clearml. [Online]. Available: https://clear.ml/docs/latest/docs/webapp/webapp_exp_table/ (Accessed 2022-04-06).

[37] T. Budras, "Evaluation of machine learning lifecycle tools in the context of a specific NLP project," Bachelor's Thesis, Department of Computer Science and Business Information Systems, University of Applied Sciences Karlsruhe, Germany, 2021. [Online]. Available: https://www.smiffy.de/thesis/thesis-buti1021.pdf (Accessed 2022-04-08).

 ISBN: 978-1-61208-969-0