sustainability

MDPI

Article

A Knowledge Graph-Based Data Integration Framework Applied to Battery Data Management

Tahir Emre Kalaycı *, Bor Bricelj, Marko Lah, Franz Pichler, Matthias K. Scharrer and Jelena Rubeša-Zrim

Virtual Vehicle Research GmbH, Inffeldgasse 21A, 8010 Graz, Austria; bor.bricelj@v2c2.at (B.B.); marko.lah@v2c2.at (M.L.); franz.pichler@v2c2.at (F.P.); matthias.scharrer@v2c2.at (M.K.S.); jelena.rubesa-zrim@v2c2.at (J.R.-Z.)

* Correspondence: emre.kalayci@v2c2.at

Abstract: Today, the automotive and transportation sector is undergoing a transformation process to meet the requirements of sustainable and efficient operations. This transformation mainly reveals itself by electric vehicles, hybrid electric vehicles, and electric vehicle sharing. One significant, and the most expensive, component in electric vehicles is the batteries, and the management of batteries is crucial. It is essential to perform constant monitoring of behavior changes for operational purposes and quickly adjust components and operations to these changes. Thus, to address these challenges, we propose a knowledge graph-based data integration framework for simplifying access and analysis of data accumulated through the operations of vehicles and related transportation systems. The proposed framework aims to enable the effortless analysis and navigation of integrated knowledge and the creation of additional data sets from this knowledge to use during the application of data analysis and machine learning. The knowledge graph serves as a significant component to simplify the extraction, enrichment, exploration, and generation of data in this framework. We have developed it according to the human-centered design, and various roles of the data science and machine learning life cycle can use it. Its main objective is to streamline the exploration and interaction with the integrated data to maximize human productivity. Finally, we present a battery use case to show the feasibility and benefits of the proposed framework. The use case illustrates the usage of the framework to extract knowledge from raw data, navigate and enrich it with additional knowledge, and generate data sets.

Keywords: anomaly detection; battery data management; data integration; data analysis; intelligent transport systems; knowledge graphs; machine learning techniques

Citation: Kalayci, T.E.; Bricelj, B.; Lah, M.; Pichler, F.; Scharrer, M.K.; Rubeša-Zrim, J. A Knowledge Graph-Based Data Integration Framework Applied to Battery Data Management. *Sustainability* **2021**, *13*, 1583. https://doi.org/10.3390/su13031583

Received: 28 December 2020
Accepted: 29 January 2021
Published: 2 February 2021

1. Introduction

In today's world, one of the prime directives is the pursuit of sustainable operations and efficiency. Especially the automotive and the transportation sector, in general, is undergoing a transformation process from the perspective of power-train technologies, driving assistance systems, as well as the whole concept of personal mobility and logistics. To achieve sustainable and efficient operations, technological components have to be engineered in a way that enables them to continuously adapt during their life cycle. More precisely, we need to ensure that they can stay deployed as long as possible and perform optimally.

Electric vehicles (EV) and hybrid electric vehicles (HEV) are the key drivers in this transformation process [1] with fuel cell (hybrid) electric vehicles (FCHEV) being around the corner [2]. A critical component of these vehicles is the batteries that are used to power them. They are one of the most expensive components [3]. The most important limitations of the commercialization of conventional battery-based EVs are the high cost and cycle life of batteries, complications of chargers, and the lack of a charging infrastructure [4]. Furthermore, battery chargers can produce harmful harmonic currents in distribution

systems [5]. The most critical non-technical factor in achieving mass-market status for a battery-based vehicle is its relative cost [6]. Despite this, EVs and HEVs promise vital results in a world where road transport is likely to remain a significant contributor to air pollution in cities [7] and is the most significant factor with nearly three-quarters of the transport-related GHG emissions [8]. Eventually, to achieve climate neutrality, a 90% reduction is necessary by 2050 in all modes of transport emission, and achieving sustainable transport means putting users first and providing them with more affordable, accessible, healthier, and cleaner alternatives to their current mobility habits [9].

Additionally, EV sharing is embraced by many cities worldwide as a flexible and sustainable means of urban transit [10]. In this respect, the appearance of transportation as a service provider is another crucial innovation driver. From a fleet management perspective, it is critical that a vehicle stays deployed as long as possible and fulfills the required tasks [11]. A significant challenge for the operators is to charge the fleet because of limited or costly access to charging facilities [10]. The information of the current battery status in the fleet can help the fleet management system to arrange the deployed vehicles according to the range that they are expected to drive. In this case, vehicles with the battery could serve customers living outside of cities, where the driving range requirement is typically higher and vehicles with a more worn-out battery would be moved to the city centers because the expected driving distance is shorter [12]. That is just one example of how to extend the life cycle of a deployed unit and contribute to more sustainable operations in modern transportation concepts.

Furthermore, a deployed battery in an electric vehicle will change its behavior over time, controlled by its on-board battery management system (BMS). These changes over time will lead to the necessity of constant monitoring of the battery for operational purposes. It is also necessary to present decision-makers a complete profile of knowledge of the effects, both financial and extra-financial, that projects (or products) are expected to produce [13,14]. Additionally, a holistic view of the transport system and all available drive technologies can provide realistic and feasible solutions and also show their ecological, economic, and social effects at a significant level, namely, for an entire urban region [15]. For example, the improvements (including cell design, factory, materials, and vehicle integration) made by the Tesla company that result in halving the cost per kWh shows us the importance of collecting data about battery efficiency and dealing with the issue from all aspects [16]. Eventually, this cost reduction helps the company to produce more affordable electric vehicles, to broaden its market to include additional buyers, and thereby reducing the number of gas-powered vehicles on the road [16].

The logical consequence of continuous monitoring is the accumulation of a large amount of data. Tanizawa et al. [17] present a cloud-connected BMS to reduce emissions by providing a data cloud that aids in battery replacement. Recently, Li et al. expand on this idea to a digital twin system with seamless data transmission [18]. Both examples work based on accumulating data in a database that is not specified in detail. Only owning and accumulating even more data does not ensure effective decision-making or any engineering advantage. What is required is a map that guides one in the vast data landscapes generated in the development process and over the entire life cycle. Knowledge graphs can act as such maps, and they can bridge different data points easily in this manner.

Constant data collection requires data integration. Data integration [19–23] can be considered as the process of combining different data that are stored in a set of separate data sources into a single unified view. However, information integration is generally understood to be the process of assembling a coherent view of the distributed heterogeneous information and information processing resources [24]. The aim of integration is generally to consolidate information wisely, free of redundancy, processed and operated by the right business logic to deliver the proper and condensed answer, and offer simple access to the end user, thereby eliminating the need for knowledge about the underlying data structures [25]. Integration also provides a way of standardizing data, improving accuracy and productivity, and greater flexibility and agility [26]. Therefore, this process is

known to be challenging in many different ways. Moreover, the challenges vary depending on the environment, and this environment typically includes a broad mix of sources, which can be either structured or unstructured [27]. Many technical factors, such as the structural and semantic heterogeneity of data sources, and the incompatibility of underlying systems in which the data sources reside, make the process more challenging [28]. In addition to this, specific non-technical challenges such as access authorization or data anonymity could be crucial in domains such as law or medicine.

The application of knowledge graphs and data integration in environmentally sustainable systems is gaining popularity in recent years due to the shift from fossil fuels to renewable energy sources. This shift requires sophisticated information systems that enable the management of those new energy sources to be reliable and cost-effective. One of those approaches is the energy internet, which was proposed by American economic and social theorist Jeremy Rifkin. The energy internet seeks to connect renewable distributed energy sources with the internet for fair access to distributed power, as well as sustainable production and consumption [29]. It is a complex network formed based on the internet and other cutting-edge information technologies. It takes distributed renewable energy as the primary energy source and is tightly coupled with natural gas networks, transportation networks, and other systems. This complex structure poses severe challenges for energy management [30–33], including complex context modeling, heterogeneous data analysis, distributed nodes integration, and dynamic controlling. The great achievement of semantic technologies obtained in the area of energy management verified the advantages of knowledge engineering in complex system scheduling and controlling. Consequently, as an improved form of knowledge representation, the knowledge graph pose a big potential in complex energy management, which consists of a large number of interconnected concepts, entities, and events [34].

Another important aspect today is the diversity of roles in data science (DS) and machine learning (ML) teams, who are working in different stages of the DS/ML life cycle. However, there are not enough DS/ML professionals to fill the job demands, and 80% of their time is generally spent on low-level activities [35] such as tweaking data [36] or trying out various algorithmic options and model tuning [37]. These two challenges have stirred artificial intelligence (AI) researchers and system builders to explore an automated solution for DS/ML work, which is called Automated Data Science (AutoML). Thus, several DS/ML algorithms and systems have been built to automate the various stages of the DS/ML life cycle [35,37].

In this respect, we propose a framework based on knowledge graph-based data integration to ease the extraction, enrichment, generation, and exploration of data sets. It aims to simplify the exploration and querying, data sets creation, and algorithm application over these generated data sets. Streamlining the exploration and interaction with the acquired data is key to maximize human productivity in the field of data science [38]. The framework is developed to address the obstacles to sustainability in the automotive and transportation sector. Therefore, the focus of the framework is on the data readiness, preprocessing, cleaning, feature engineering, model building and training, and model refinement stages of the DS/ML life cycle.

The remainder of this paper is organized as follows. We present the proposed framework in Section 2; we describe the battery use case in Section 3; and finally, we conclude the paper in Section 4 with discussions, conclusions, and future work.

2. Methodology: Knowledge Graph-Based Data Integration Framework

In this section, we describe our knowledge graph-based data integration framework that addresses the needs of various systems utilizing data science and machine learning techniques. The main objective of the framework is to enable the users to access and navigate the data seamlessly and to apply data science and machine learning techniques without any difficulty. We eventually want to standardize the data, improve accuracy and productivity, and provide greater flexibility and agility coherent to the aims of integration.

From the user perspective, we want to create transparent and unified access to diverse and heterogeneous data sources.

In this respect, we store domain knowledge in a schema-free Neo4j graph database [39] as a knowledge graph, which is accessible through a single access layer to ensure the achievement of these goals.

We describe the proposed framework in a logical bottom-up direction, starting from data intake and ending with the user interaction with it. In this regard, our framework consists of different components (Figure 1) to meet the requirements explained above. We will describe each of these components individually in the following sections following this bottom-up narrative.

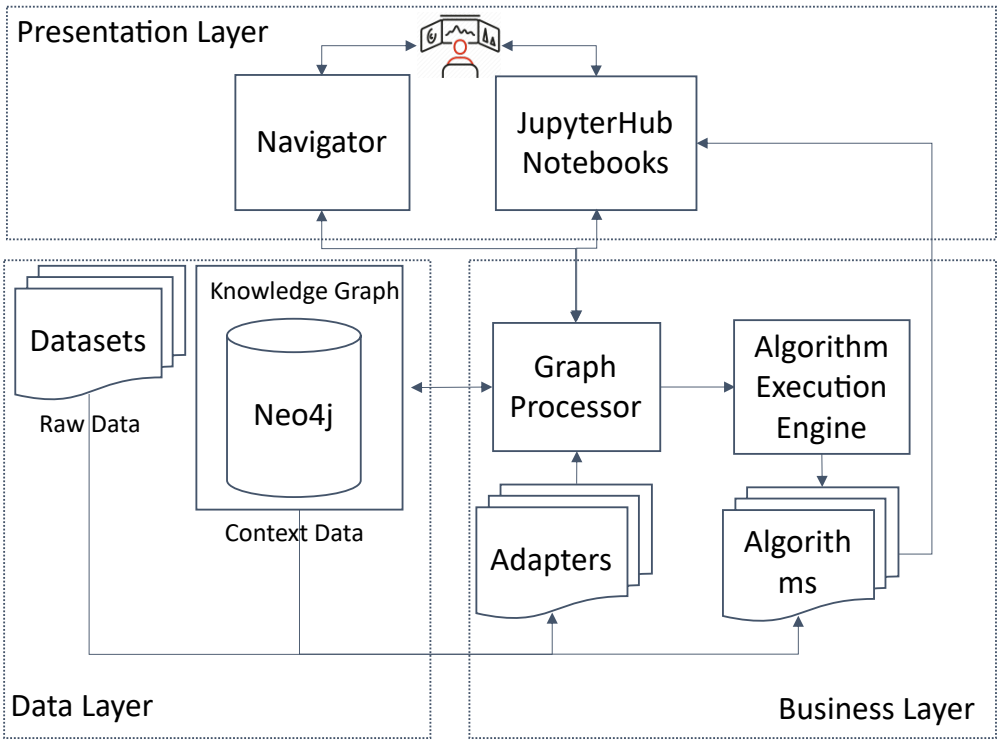

Figure 1. Components of the proposed framework and their communication.

2.1. Storing the Data: Knowledge Graph

A Knowledge Graph (KG) is a self-descriptive knowledge base where data and its schema are stored in a graph format, and the relations between data are first-class citizens [40]. It consists of a set of interconnected typed entities and their attributes [41]. They are distinguished from other knowledge-oriented information systems by their particular combination of [42]

(i) knowledge representation structures and reasoning, such as languages, schemas, standard vocabularies, and hierarchies among concepts;
(ii) information management processes (how information is ingested, transformed into a knowledge graph); and
(iii) accessing and processing patterns, such as querying mechanisms, search algorithms, and pre- and postprocessing techniques.

It is known as a triple-based data model or a property model [43] and various modern applications widely use the KGs to virtually integrate data from diverse sources on a single database [33]. Algorithms that run on top of KGs include inference, recommendations, machine learning, and text understanding, to name a few [40]. Eventually, there is increasing usage of KGs in different applications because of these reasons. Our main reason is to benefit the integrating various sources into a single view and easily accessible database.

We employ the labeled property graph [44] approach using the Neo4j Graph Database. The purpose of using a graph database originates from the performance, flexibility, and agility advantages of them [39]. Graph databases have become an important solution to consider in the management of large datasets [45]. Using them to manage graph-structured data presents many benefits such as explicit support for modeling graph data, native indexing and storage for fast graph traversal operations, built-in support for graph algorithms (e.g., Page Rank and subgraph matching), and the provision of graph languages, allowing users to express complex pattern-matching operations [46,47]. The main characteristic of a graph database is that the data are conceptually modeled and presented to the user as a graph [48]. The data structures (data and/or schema) are represented by graphs, or by data structures generalizing the notion of the graph (e.g., hypergraphs or hyper nodes). One of the main features of a graph structure is the simplicity to model unstructured data. Therefore, in graph models, the separation between schema and data is less explicit than the classical relational model.

Both the notion of Property Graphs (PG) and the Resource Description Framework (RDF) are commonly used models for representing graph-shaped data [49]. However, we specifically focus on labeled property graphs because we have decided to use Neo4j as the graph database. Informally, a property graph is a directed labeled multi-graph with the special characteristic that each node or edge could maintain a set (possibly empty) of property–value pairs [48].

- The labeled property graph [39] contains nodes, relationships, properties, and labels.
- Nodes contain properties. It is possible to tag nodes with one or more labels. Labels indicate the roles that nodes play within the data set.
- Relationships connect nodes and structure the graph. They are named, directed, and always have a start and an end node. Name and direction provide semantic clarity to the structure.
- Relationships also contain properties, which is useful to provide additional metadata for graph algorithms, adding additional semantics, and for constraining queries at run-time.
- Properties are in the form of arbitrary key–value pairs. Keys are usually strings, and values are arbitrary data types.

Consequently, Neo4j can be used as a high-performance replacement for relational databases, especially when handling highly interconnected data [50]. We use the Cypher query language, which was specific to the Neo4j graph database, but currently, many other different commercial products are also using it [46]. It is a well-established declarative language for querying and updating property graph databases [47]. It is an expressive language for property graphs and is designed to be easily read and understood by developers and database professionals. SQL influenced the design of Cypher, but how the language operates is relatively different from SQL [51]. A Cypher statement is a sequence of clauses: the next clause will take its input from the result of the previous clause. It is also possible to freely combine static and dynamic features. One can start with a pattern-matching clause, then update a database based on the matched patterns, and then proceed with another pattern-matching clause on the updated database—all in one single statement [51]. For these advantages, we are storing the domain-specific data as a KG in the Neo4j.

The proposed framework aims the modularity as much as possible and each component in the architecture aims to meet this. It provides the users to access the domain data through the KG, which consists of the processed data in the Neo4j and raw data likely possibly stored in other data sources such as file system.

2.2. Managing and Accessing the Domain-Specific Data: Graph Processor

The central component in the architecture is the *Graph Processor*, which essentially deals with accessing the KG that the Neo4j graph database stores without a particular schema. This component acts as the back end and exclusive interface for accessing the KG.

This central component implements the following tasks for the framework:

- processes the raw domain data and imports the necessary parts into the KG using the specific adapter,
- acts as the back end and provides access of different components to the KG,
- acts as the back end for triggering algorithm execution engine (Section 2.3).

The main task of this component is creating the KG using adapters, which transforms the raw data into the Neo4j graph database. In this regard, adapters are essential to store the data in the graph and to support access to the data sets. Adapters facilitate the loading of different data sets into the KG smoothly. They provide the general means of supporting any domain data without changing anything specific in the framework. They also provide the means to access files and/or data that are not stored in the KG directly for storage and performance reasons. These files can be in the file system, and the specific data adapter provides access to them. Expert users must develop the adapters required to create the KG. Then, these adapters are stored in the framework to be loaded with parameters when they are needed. They are loaded during run-time by the request of users and they are linked during run-time rather than compilation [52].

We provide some standard adapters (such as JSON, CSV, and relational database) with the implementation. We assume that expert users, who know the domain data, can develop additional custom adapters.

2.3. Executing the Algorithms: Algorithm Execution Engine

The *Algorithm Execution Engine* (AEE) component (Figure 2) provides the means of running different algorithm modules that are provided by users. The spawned notebook server of the user and remote workers that are provided with the deployment of the framework constitute the AEE. Users can easily upload or develop algorithm modules in their server, run them in the notebook server during development to test and improve, and finally can send these to a provided remote worker for asynchronous execution.

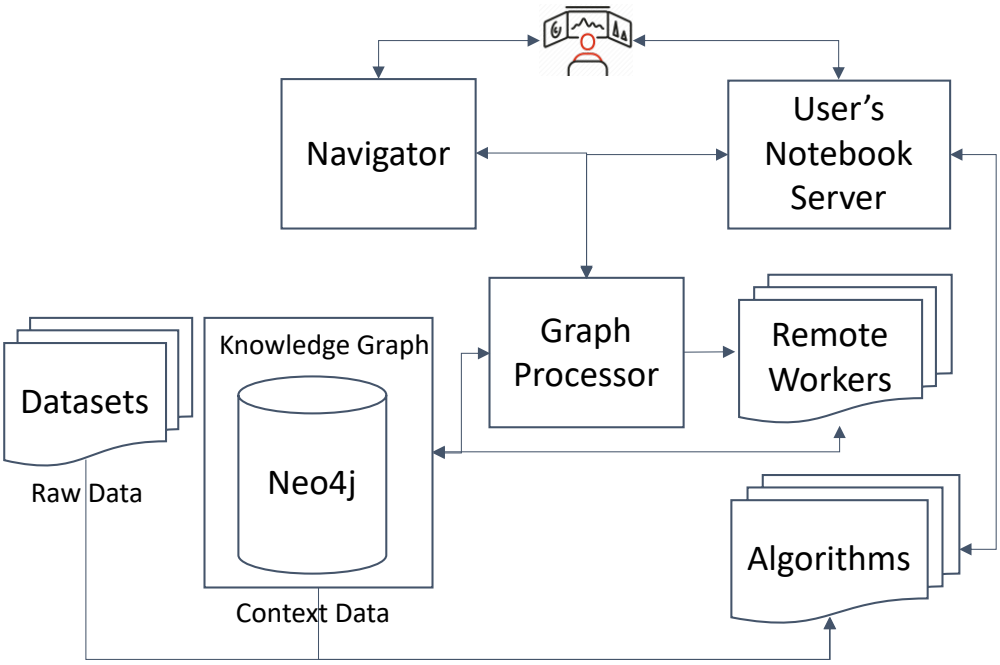

Figure 2. The Algorithm Execution Engine consists of user's notebook server and remote workers to run algorithms.

The algorithm modules provided by the user can perform different tasks:

- executing enrichment algorithms to extend the KG from different public–private APIs, such as weather, geographic position, map, point of interest;
- executing various data analysis techniques; and

- executing various machine learning techniques.

All algorithm modules can use the libraries provided in the custom Jupyter image that is used to create the user's Jupyter notebook server container and can use the complete or a subset of KG as the data-set.

The AEE architecture allows the users to provide any kind of algorithm with ease and simplifies the application of these algorithms to the KG. It is important to note that the algorithms and adapters can be distributed and executed asynchronously using task queues or remote workers (e.g., Celery, Redis Queue, Taskmaster, Dramatiq, IronWorker, etc.). That is especially the case when they take a long time to run and/or a single computer will not be enough as a computation resource. This mechanism also allows distributed and cloud computing for scalability. These workers can run on different on-premise computers or cloud providers when needed and algorithms can be distributed over these computing resources with the help of AEE.

We assume that there are some standard algorithms (such as exploratory data analysis, statistical analysis, simple aggregation, and analysis) that are provided with the implementation, and expert users that know the details can develop additional custom algorithms for the users.

2.4. Interaction with the System: Notebooks and Navigator

The *Notebooks* component provides a JupyterHub server to allow the users to interact and run different algorithms, adapters, and similar notebooks in separate Docker containers by accessing the KG. JupyterHub [53] is a tool that aims to serve Jupyter notebook [54] for multiple users. Jupyter is a free, open-source, and interactive web tool known as a computational notebook. Researchers can use it to combine software code, computational output, explanatory text, and multimedia resources in a single document. It has emerged as a de facto standard for data scientists and aims to foster computational reproducibility by simplifying the code reuse [55]. For these reasons, we are using JupyterHub as a multi-user Hub to spawn, manage, and proxy multiple instances of the single-user Jupyter notebook server. It consists of

(i) a Hub (tornado process) that is the heart of JupyterHub,
(ii) a configurable HTTP proxy that receives the requests from the client's browser,
(iii) multiple single-user Jupyter notebook servers that are monitored by Spawners,
(iv) and an authentication class that manages how users can access the system.

We implemented a custom authentication system attached to use the same authorization with the navigator component and we use DockerSpawner [56] to enable JupyterHub to spawn independent single-user notebook servers in Docker containers. Our JupyterHub component takes an authenticated user and spawns a notebook server in a Docker container for that authenticated user using a custom Jupyter notebook image extending Jupyter Notebook Scientific Python Stack with Tensorflow [57]. The user can perform operations in this notebook container over the KG accessed through GP. The container allows the user to execute the algorithms with and without interaction either in remote workers or in itself.

The final component is the *Navigator* component, which provides the means for data navigation to the end-user. It is designed as a simple dashboard for the user to easily navigate the KG, filter out required subsets to provide them to the notebook server of the user for the algorithms.

2.5. Implementation

We have created a modular implementation, and each component is individually implemented based on the concepts discussed previously. We keep them isolated and build loosely coupled services deployed as Docker containers. Docker simplifies and accelerates the workflow while giving the freedom to innovate with a custom choice of tools, application stacks, and deployment environments for each [58]. By using Docker, components are isolated from each other and bundle their software, libraries, and configuration files [58].

The components communicate with each other through RESTful methods, which ensure interoperability between them. Architectural constraints provided through REST emphasizes scalability of component interactions, the generality of interfaces, and independent deployment of components [59].

As each component developed individually, we selected programming language and tools that are suitable to the task of the component:

- Graph Processor (Section 2.2) implemented using Python programming language and Flask web framework [60] with Flask-RESTful [61] extension to provide a RESTful API.
- Algorithm Execution Engine (Section 2.3) is implemented using Python programming language to support open-source Celery asynchronous task queue [62]. However, algorithms and adapters can be implemented using Python programming language and other programming languages that are supported in the Jupyter notebook server and the deployment.
- We developed our notebooks (Section 2.4) component by extending the default JupyterHub Docker image to use DockerSpawner, a custom authenticator, and the custom Jupyter notebook image.
- Navigator (Section 2.4) is implemented using open-source JavaScript library React [63]. React is a library for developing composable user interfaces and promotes the creation of reusable user interface (UI) components, which present data changes over time [64]. It is designed to enhance interactive UI development by making it easier to update the view when the data changes [63]. The view is divided into smaller components, that can be composed to create complex UIs. Components are built-in JavaScript instead of templates, enabling an easy flow of data [63].

Eventually, the proposed architecture and its implementation meet the requirements of having a modular architecture and address some important design principles:

- Each component respects the separation of concerns principle and focus on tasks related to that specific concern they are dealing with. Therefore, the proposed architecture conforms to modularity as a paradigm of separation of concerns.
- Components are loosely coupled and communicate through standardized interfaces via RESTful APIs.
- The deployment is simplified because of using individual Docker containers for each component.
- System as a whole hides the details from end-user while distinguishing between expert users that develop the adapters and algorithms, and end users that interact with the system directly through the presentation layer only.

3. Results: Using the Framework in a Battery Use Case

In this section, we describe the battery use case to demonstrate and validate the feasibility and applicability of the proposed knowledge graph-based data integration framework (Section 2). With this use case, we aim to illustrate the feasibility of the framework in various scenarios, spanning different knowledge domains. We specifically focus on providing an anomaly detection component, which is utilized directly during the graph enrichment process. Therefore, first, we describe the battery use case (Section 3.1), and then we present the usage of the framework for this use case step by step to validate the feasibility and applicability of the proposed framework (Section 3.2).

3.1. Battery Use Case

During the battery charge/discharge cycles, data is collected on various sensory observations such as capacity, current, temperature, loading difference, etc. Some of the collected sensory readings could potentially exhibit anomalous behavior, indicating either problem with the telemetry collection process or problems with the tested battery cells.

An example of such anomalous behavior is the detection of temperature spikes during discharge cycles. In that case, we pose the following hypothesis:

Hypothesis 1. *Slight temperature fluctuations during a* ***battery discharge process*** *may exist, but the* ***bigger fluctuations with extreme temperature spikes*** *could be* ***anomalous****.*

Example 1. *During experiments at room temperature, it was observed that the* ***battery temperature*** *during the* ***discharge process*** *was around* ***25 °C*** *on average.*

Remark 1. *Therefore, an observed* ***temperature spike****, i.e., a local maximum in the temperature measurement data, towards* ***50 °C*** *or more can be considered* ***anomalous****.*

The solution for the anomaly detector consisted of conducting an exploratory data analysis (EDA) and executing the following steps:

1. Detect discharge cycles.
2. Detect potential anomalous temperature spikes.
3. Extract temperature spikes and interval data.

Exploratory data analysis includes steps to identify fluctuations in the data, possible missing values, or breaks in the data, ensuring that the data is in proper format and available for subsequent analysis.

After data ingestion, the anomaly detection algorithm identifies the potentially anomalous temperature spikes using relative extrema identifying approach with a fixed window for comparison. The technique serves to identify and record the local temperature maxima, i.e., temperature spikes. The threshold for the identified spikes is defined as one being more than ten standard deviations above the mean temperature of the data sample ($t_i \geq \bar{t} + 10 * \sigma$, where t is temperature), thus ensuring a relative measure in regards to possible changes in testing conditions.

After detecting and filtering the potentially anomalous spikes, the interval boundaries for discharge cycles where the spikes occur are calculated using Algorithm 1.

Upon identification of intervals and temperature spikes, the extracted data is used to enrich the data samples in the KG.

3.2. Extract, Enrich, Explore, and Generate

A two-stage process is required to generate an enriched knowledge graph (KG) from the time-series data that are stored in a relational database.

In the first stage, the relational schema of the SQL database, which contains the raw measurement data, serves as the basis for the initial KG generation (Figure 3). The dedicated adapter for relational databases can extract the nodes and properties based on the database schema together with the defined foreign key constraints, which constitute the relationships between the extracted nodes.

In the second stage of the process, the Algorithm Execution Engine (AEE) component starts the work using the time-series data stored in the source relational database as input (Figure 4). The AEE component can communicate with the relational adapter to retrieve the required information to perform the data analysis steps explained in Section 3.1. AEE can contain multiple implemented algorithms communicating with one or multiple installed adapters in the system to perform individual tasks related to the enrichment of the graph.

We are focused on detecting possible temperature outliers during discharges in the measured battery cells in the use case. The implemented algorithm identifies local temperature maximums and the corresponding data points in the measurements that represent the relevant context (Figure 5).

After the detection of local maximums, the algorithm extracts the potential temperature outliers and the relevant intervals (Figure 6). The identified intervals are translated into new nodes that will be integrated into the initial KG. The newly generated nodes are equipped with a label and relevant properties that represent the required information to identify the particular data-points in the source data. This integration process of new nodes represents the core of the enrichment process of the KG.

Algorithm 1 Calculation of interval boundaries of discharge cycles

data: the data frame containing all records
lead: the number of data points
spikes: the list of temperature spikes

procedure FIND-INTERVAL-START($data, lead, spikes$)
    $interval_{start} \leftarrow \emptyset$ ▷ find and store starting timestamps of intervals
    for all $spike \in spikes$ **do**
        for all $i \in [spike.index, 0)$ **do**

            $dq_{sum} \leftarrow \sum_{k=i-lead}^{i} data[k].dq$ ▷ dq: discharge quantity

            if $dq_{sum} = 0$ **then**
                $interval_{start} \leftarrow interval_{start} \cup \{data[i].timestamp\}$
                break
            end if
        end for
    end for
    return $interval_{start}$
end procedure

procedure FIND-INTERVAL-END($data, lead, spikes$)
    $interval_{end} \leftarrow \emptyset$ ▷ find and store ending timestamps of intervals
    for all $spike \in spikes$ **do**
        for all $i \in [spike.index, data.length)$ **do**

            $dq_{sum} \leftarrow \sum_{k=i}^{i+lead} data[k].dq$

            if $dq_{sum} = 0$ **then**
                $interval_{end} \leftarrow interval_{end} \cup \{data[i].timestamp\}$
                break
            end if
        end for
    end for
    return $interval_{end}$
end procedure

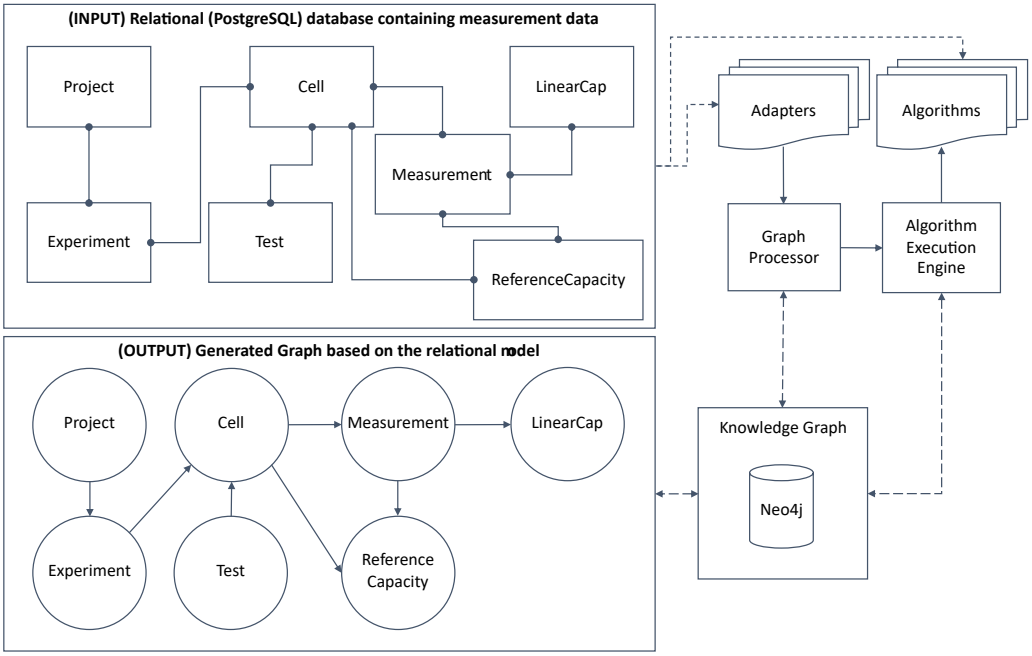

Figure 3. Representation of the workflow for extracting a base graph from the schema of the relational database containing the raw measurement data.

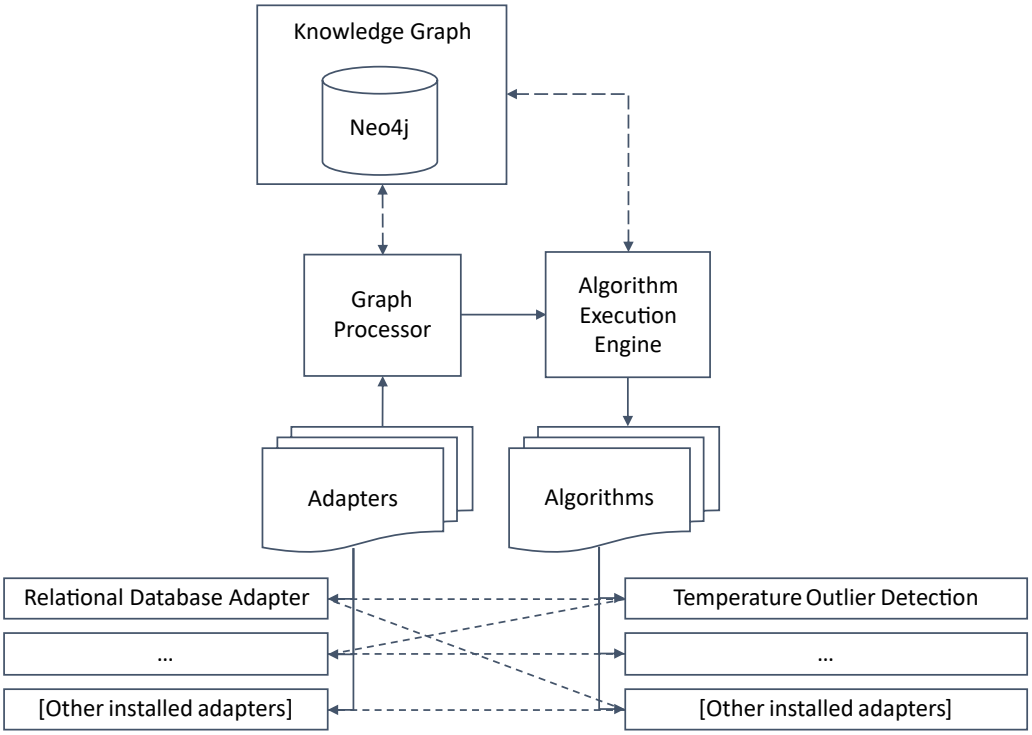

Figure 4. Adapter and Algorithm Execution Engine integration.

After the second stage, we will have a graph structure that contains the basic information schema of the source data enriched with the outcomes of the algorithm execution performed by the modules installed in the AEE component (Figure 7).

In our use case, the additional nodes represent the relevant temperature spikes during discharges in the measured battery cells. The created nodes include the properties which enable the graph processing engine and the adapter to retrieve the particular data point with the corresponding context. The advantage of holding this data in a graph structure is the ability for rapid exploration and analysis.

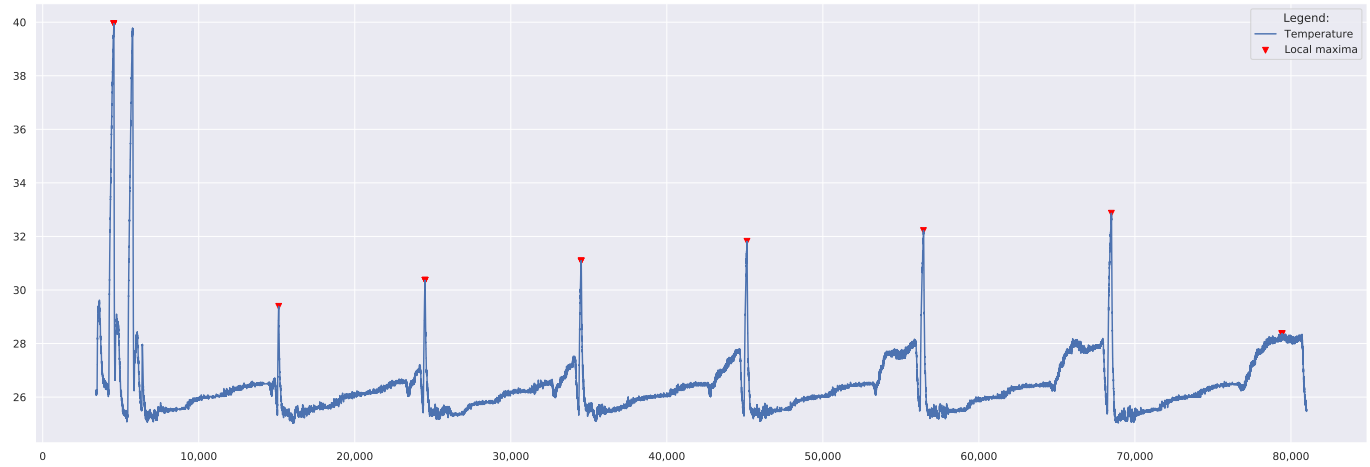

Figure 5. Temperature plot with local maxima, which indicate the temperature spikes.

Figure 6. Temperature (left y-Axis) and dq (right y-Axis) plot with local maxima outliers and discharge interval.

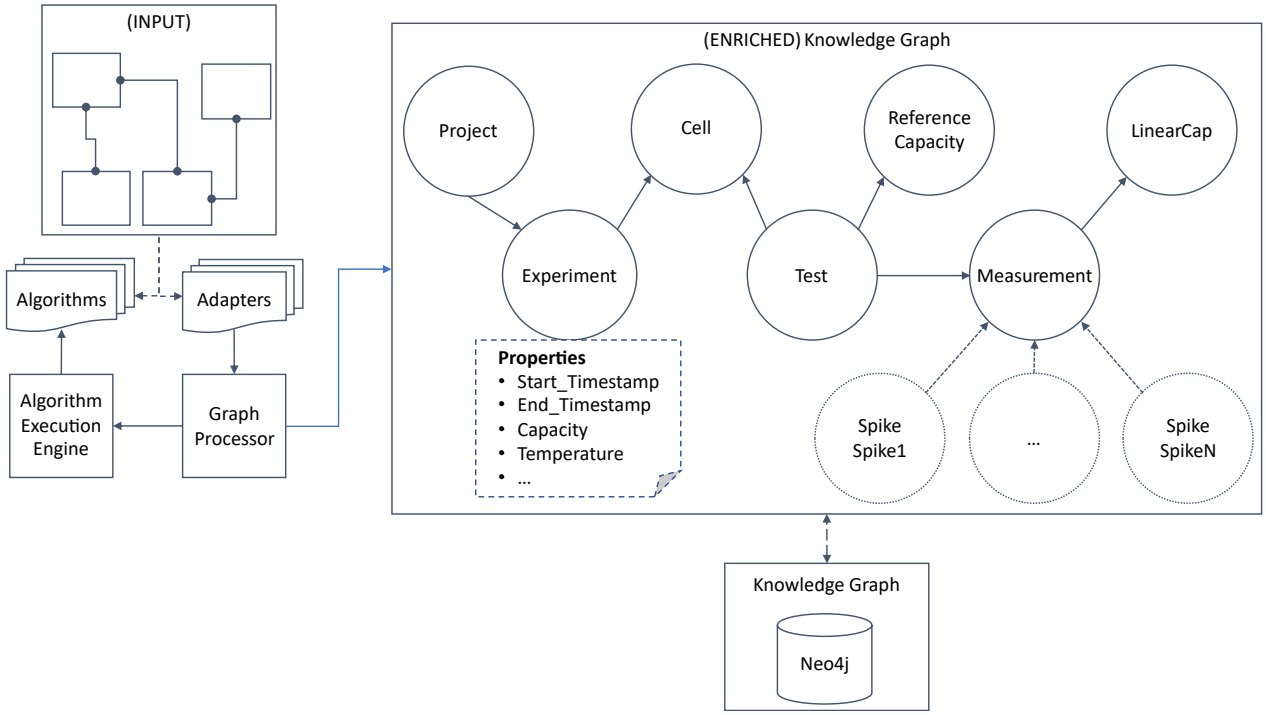

Figure 7. Graph state after the enrichment process.

Rapid exploration includes the possibility for the user to navigate the source data with the support of the KG. The advantage of using a graph structure is the ability to query for paths between individual data points. If we take the example from our use case (Figure 3) we see that there is an indirect connection between a data point or record from the `Measurement` and `Experiment` tables. With the traditional approach in the relational database domain, the user needs to know all the tables and relationship definitions that are required to traverse between these tables and create a Join. Storing this information in the graph helps the adapter component to query the shortest path, get the required information, and generate the shortest possible join or a navigation path that includes a particular set of entities that should be included along the way. Based on the path information between two data points in the graph, the adapter component can generate a corresponding SQL query that can retrieve the actual data in a tabular format (Figure 8).

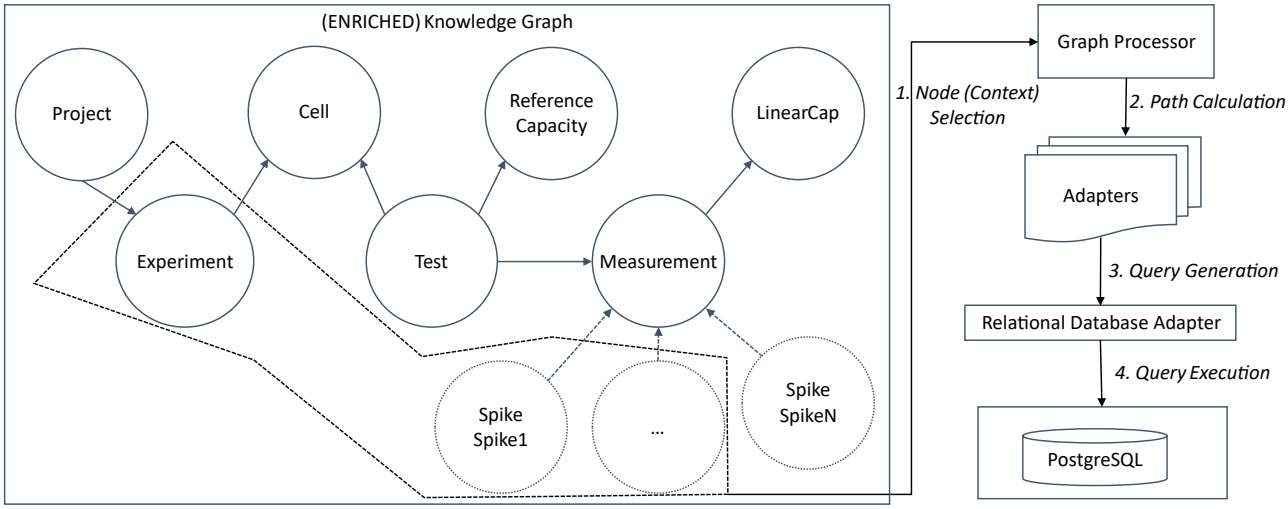

Figure 8. Abstract interaction process between a KG and a client (user/machine).

The client does not need detailed knowledge about the *structure* or *schema* of the data. The knowledge of how entities in the data are related to each other is stored into the graph via the two-staged import and enrichment process performed by the *Relational Adapter*, *Graph Processor*, and *Algorithm Execution Engine* components. The manifestation of this information in the graph database defines a KG. One potential client for using the KGs and its ecosystem can be a human user. Usually, a data scientist or analyst is performing different context selection (Figure 8) and extracting the required data from the raw database.

The second potential client could be another machine/information system that can pull the data based on a predefined context/node selection (Figure 9). Both types of clients will benefit from the graph structure as the context is resilient to structural changes. If a particular path between two entities is changed (renamed, added, removed, etc.), a route can still be calculated between two particular data points and the required information can still be retrieved. That is true for cases where the client does not depend on the particular information that was added as part of the route between data points. However, also in the case where a structural change causes a certain calculated path and consequently a query to be invalid, the investigation and resolution can be accelerated with the help of the use of the knowledge graphs.

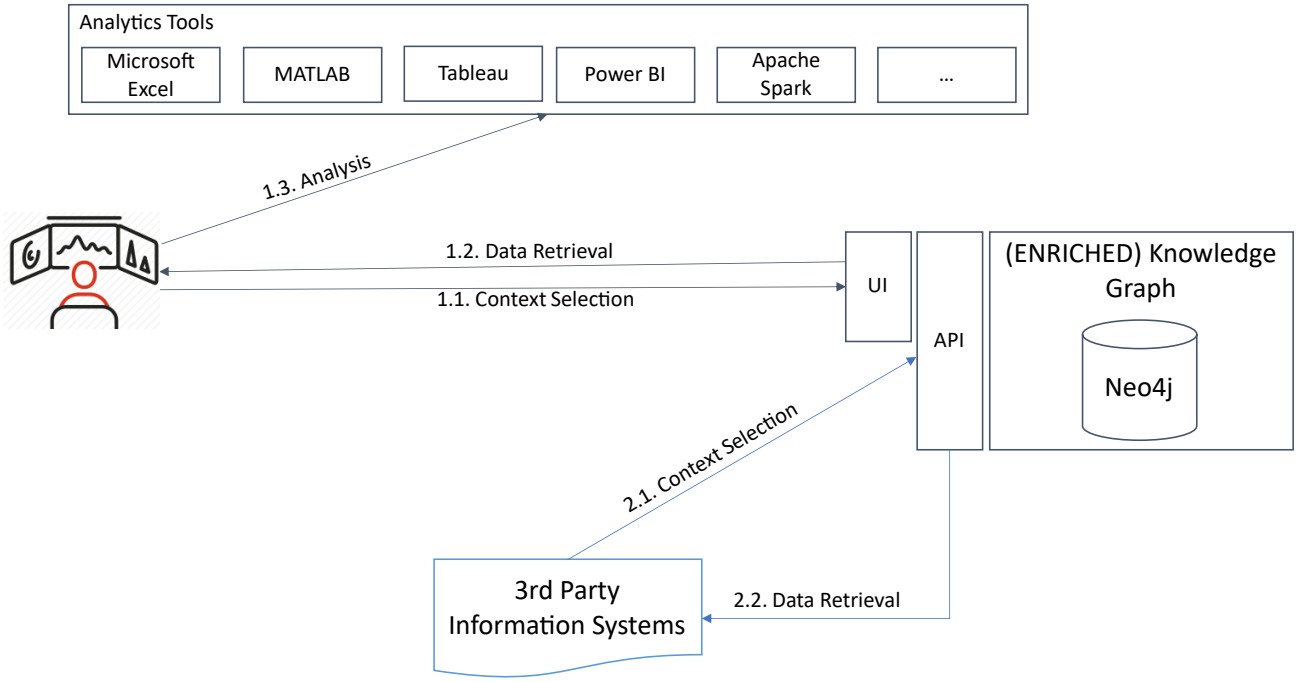

Figure 9. Abstract interaction process between a knowledge-graph an a client (user/machine).

4. Discussion and Conclusions

Knowledge graphs and graph databases, in general, facilitate the relationship between data points. They allow the user to pose queries that aim to link different data sets, which could be related to each other. Additionally, they provide the means to navigate a data warehouse smoothly. The knowledge part in a knowledge graph is added during the Extract–Transform–Load (ETL) process where data is extracted from the source, transformed and integrated into the available data based on rules, and finally loaded into graph database [65]. The transformation and rule-based integration are the key mechanisms of how to manifest a specific know-how/knowledge into a data structure accessible to other users or systems. There is a particular lack of tooling and concepts in the area of data exploration for a more traditional data warehouse or modern data lakes environments in various domains [66–68]. With the growing data source diversity and schema complexity, the ability to look at the data from different perspectives is essential. To achieve that, we need to contextualize entities and data points [69]. Therefore, the knowledge graph, built on a native graph database, serves as a crucial component by helping the user to navigate over a particular set of directly or indirectly related data points, creating the possibility of easier querying and smooth creation of data sets used in advanced data analysis and machine learning use cases.

In this context, our proposed framework intends to provide the benefits of knowledge graphs by providing clear means to analyze and navigate knowledge stored in KG. The extraction, enrichment, exploration, and generation steps of the proposed framework deliver a seamless platform to the users. We present a battery use case to show and validate the feasibility and applicability of the proposed framework. It demonstrates how a user can easily navigate the data that is extracted from the raw data source with the help of KG and how they can enrich it with extra knowledge produced directly from raw data or obtained from other sources. Finally, we demonstrate how users can generate additional data sets to directly or indirectly use in the subsequent data analysis and machine learning tasks. Consequently, our use case validates how users can employ the framework for the application of data analysis and machine learning approaches in a clear, smooth, and context-aware way.

Therefore, our proposed framework is an L1 Human-directed automation tool with enrichment capabilities according to the theoretical framework for Human-in-the-Loop

AutoML suggested by Wang et al. [35]. It is developed according to the human-centered design and it is meant to be used by various roles, who are especially focused on *Data Readiness and Preparation*, *Feature Engineering*, *Model Building and Training*, and *Model Refinement* stages of the life cycle.

In the future, we want to implement additional components to cover and simplify the *Data Acquisition and Governance*, *Model Deployment*, and *Model Monitoring* stages and improve the support of multi-user communications and collaborations in every stage and across different stages. Finally, as the relationship between user trust and AutoML systems is crucial to ensure that these tools can be trustworthy enough to be adopted responsibly by the public [70], we also plan to investigate the explainability and trust to allow the users to understand the results and automation better.

Author Contributions: Conceptualization, T.E.K., M.L., and J.R.-Z.; data curation, B.B. and M.K.S.; funding acquisition, M.L. and J.R.-Z.; investigation, B.B., M.L., and M.K.S.; methodology, T.E.K., B.B., and M.L.; project administration, M.L. and J.R.-Z.; software, T.E.K., B.B., and M.L.; visualization, T.E.K., B.B., and M.L.; writing—original draft, T.E.K., B.B., and M.L.; writing—review and editing, T.E.K., F.P., M.K.S., and J.R.-Z. All authors have read and agreed to the published version of the manuscript.

Funding: Virtual Vehicle Research GmbH has received funding within COMET Competence Centers for Excellent Technologies from the Austrian Federal Ministry for Climate Action, the Austrian Federal Ministry for Digital and Economic Affairs, the Province of Styria (Dept. 12) and the Styrian Business Promotion Agency (SFG). The Austrian Research Promotion Agency (FFG) has been authorized for the program management.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Pelegov, D.V.; Pontes, J. Main Drivers of Battery Industry Changes: Electric Vehicles—A Market Overview. *Batteries* **2018**, *4*, 65. [CrossRef]
2. Das, H.S.; Tan, C.W.; Yatim, A. Fuel cell hybrid electric vehicles: A review on power conditioning units and topologies. *Renew. Sustain. Energy Rev.* **2017**, *76*, 268–291. [CrossRef]
3. The Volkswagen Group. Lithium to Lithium, Manganese to Manganese. 2019. Available online: https://www.volkswagenag.com/en/news/stories/2019/02/lithium-to-lithium-manganese-to-manganese.html (accessed on 20 January 2021).
4. Jayakumar, A.; Chalmers, A.; Lie, T.T. Review of prospects for adoption of fuel cell electric vehicles in New Zealand. *IET Electr. Syst. Transp.* **2017**, *7*, 259–266. [CrossRef]
5. Beretta, J. *Automotive Electricity: Electric Drives*; John Wiley & Sons: Hoboken, NJ, USA, 2013. [CrossRef]
6. Tran, M.; Banister, D.; Bishop, J.D.K.; McCulloch, M.D. Realizing the electric-vehicle revolution. *Nat. Clim. Chang.* **2012**, *2*, 328–333. [CrossRef]
7. World Health Organization. *Health Effects of Transport-Related Air Pollution*; World Health Organization Europe: Copenhagen, Denmark, 2005.
8. Climate Change—Driving Forces. 2020. Available online: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Climate_change_-_driving_forces (accessed on 20 January 2021).
9. European Commission. Communication and Roadmap on the European Green Deal. 2019. Available online: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52019DC0640 (accessed on 20 January 2021).
10. He, L.; Ma, G.; Qi, W.; Wang, X. Charging an Electric Vehicle-Sharing Fleet. *Manuf. Serv. Oper. Manag.* **2019**. [CrossRef]
11. Nuhic, A.; Bergdolt, J.; Spier, B.; Buchholz, M.; Dietmayer, K. Battery Health Monitoring and Degradation Prognosis in Fleet Management Systems. *World Electr. Veh. J.* **2018**, *9*, 39. [CrossRef]
12. Chen, S.; Andrienko, G.; Andrienko, N.; Doulkeridis, C.; Koumparos, A. Contextualized Analysis of Movement Events. In Proceedings of the EuroVis Workshop on Visual Analytics (EuroVA 2019), Porto, Portugal, 3 June 2019; The Eurographics Association: Geneva, Switzerland, 2019. [CrossRef]
13. Nesticò, A.; Maselli, G. Declining DiscountRate Estimate in the Long-Term Economic Evaluation of Environmental Projects. *J. Environ. Account. Manag.* **2020**, *8*, 93–110. [CrossRef]

14. Gangolells, M.; Casals, M.; Forcada, N.; Macarulla, M. Life Cycle Analysis of a Game-Based Solution for Domestic Energy Saving. *Sustainability* **2020**, *12*, 6699. [CrossRef]
15. Göhlich, D.; Nagel, K.; Syré, A.M.; Grahle, A.; Martins-Turner, K.; Ewert, R.; Miranda Jahn, R.; Jefferies, D. Integrated Approach for the Assessment of Strategies for the Decarbonization of Urban Traffic. *Sustainability* **2021**, *13*, 839. [CrossRef]
16. Frazelle, J. Battery Day: A Closer Look at the Technology That Makes Portable Electronics Possible. *Queue* **2020**, *18*, 5–25. [CrossRef]
17. Tanizawa, T.; Suzumiya, T.; Ikeda, K. Cloud-connected battery management system supporting e-mobility. *Fujitsu Sci. Tech. J.* **2015**, *51*, 27–35.
18. Li, W.; Rentemeister, M.; Badeda, J.; Jöst, D.; Schulte, D.; Sauer, D.U. Digital twin for battery systems: Cloud battery management system with online state-of-charge and state-of-health estimation. *J. Energy Storage* **2020**, *30*, 101557. [CrossRef]
19. Lenzerini, M. Data Integration: A Theoretical Perspective. In Proceedings of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'02), Madison, WI, USA, 3–5 June 2002; ACM: New York, NY, USA, 2002; pp. 233–246. [CrossRef]
20. Noy, N.F. Semantic Integration: A Survey of Ontology-based Approaches. *SIGMOD Rec.* **2004**, *33*, 65–70. [CrossRef]
21. Alexiev, V.; Breu, M.; Bruijn, J.D.; Fensel, D.; Lara, R.; Lausen, H. (Eds.) *Information Integration with Ontologies: Experiences from an Industrial Showcase*; John Wiley & Sons: Hoboken, NJ, USA, 2005.
22. Fagin, R.; Kolaitis, P.G.; Miller, R.J.; Popa, L. Data exchange: Semantics and query answering. *Theor. Comput. Sci.* **2005**, *336*, 89–124. [CrossRef]
23. Doan, A.; Halevy, A.; Ives, Z. *Principles of Data Integration*, 1st ed.; Morgan Kaufmann Publishers: San Francisco, CA, USA, 2012.
24. Hung, P.C.K.; Chiu, D.K.W. Developing Workflow-Based Information Integration (WII) with Exception Support in a Web Services Environment. In Proceedings of the IEEE 37th Annual Hawaii International Conference on System Sciences (HICSS 2004), Big Island, HI, USA, 5–8 January 2004; [CrossRef]
25. Maier, A.; Schnurr, H.P.; Sure, Y. Ontology-Based Information Integration in the Automotive Industry. In Proceedings of the 2nd International Semantic Web Conference (ISWC'03), Sanibel Island, FL, USA, 20–23 October 2003; Springer: Berlin/Heidelberg, Germany, 2003; pp. 897–912. [CrossRef]
26. Xu, L.D. *Enterprise Integration and Information Architecture*, 1st ed.; Auerbach Publications: New York, NY, USA, 2014. [CrossRef]
27. Haas, L. Beauty and the Beast: The Theory and Practice of Information Integration. In Proceedings of the 11th International Conference on Database Theory (ICDT'07), Barcelona, Spain, 10–12 January 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 28–43. [CrossRef]
28. Kalayci, T.E.; Güzel Kalayci, E.; Lechner, G.; Neuhuber, N.; Spitzer, M.; Westermeier, E.; Stocker, A. Triangulated investigation of trust in automated driving: Challenges and solution approaches for data integration. *J. Ind. Inf. Integr.* **2021**, *21*, 100186. [CrossRef]
29. Rifkin, J. *The Third Industrial Revolution: How Lateral Power Is Transforming Energy, the Economy, and the World*; St. Martin's Press: New York, NY, USA, 2011.
30. Anzaldi, G.; Corchero, A.; Wicaksono, H.; McGlinn, K.; Gerdelan, A.; Dibley, M.J. Knoholem: Knowledge-Based Energy Management for Public Buildings Through Holistic Information Modeling and 3D Visualization. In Proceedings of the 2nd International Technology Robotics Applications Conference (INTERA), Oviedo, Spain, 11 March 2013; Springer: Cham, Switzerland, 2014; pp. 47–56.
31. Lee, D.; Cheng, C.C. Energy savings by energy management systems: A review. *Renew. Sustain. Energy Rev.* **2016**, *56*, 760–777. [CrossRef]
32. Fan, C.; Song, M.; Xiao, F.; Xue, X. Discovering Complex Knowledge in Massive Building Operational Data Using Graph Mining for Building Energy Management. *Energy Procedia* **2019**, *158*, 2481–2487. [CrossRef]
33. Chun, S.; Jung, J.; Jin, X.; Seo, S.; Lee, K.H. Designing an integrated knowledge graph for smart energy services. *J. Supercomput.* **2020**, *76*, 8058–8085. [CrossRef]
34. Wang, X.; Ma, C.; Liu, P.; Pan, B.; Kang, Z. A Potential Solution for Intelligent Energy Management-Knowledge Graph. In Proceedings of the 2018 IEEE International Conference on Energy Internet (ICEI), Beijing, China, 21–25 May 2018; IEEE: Los Alamitos, CA, USA, 2018; pp. 281–286. [CrossRef]
35. Wang, D.; Liao, Q.V.; Zhang, Y.; Khurana, U.; Samulowitz, H.; Park, S.; Muller, M.; Amini, L. How Much Automation Does a Data Scientist Want? *arXiv* **2021**, arXiv:2101.03970.
36. Heer, J.; Shneiderman, B. Interactive Dynamics for Visual Analysis. *Commun. ACM* **2012**, *55*, 45–54. [CrossRef]
37. Zöller, M.A.; Huber, M.F. Benchmark and Survey of Automated Machine Learning Frameworks. *arXiv* **2021**, arXiv:1904.12054.
38. Crotty, A.; Galakatos, A.; Zgraggen, E.; Binnig, C.; Kraska, T. The Case for Interactive Data Exploration Accelerators (IDEAs). In Proceedings of the Workshop on Human-In-the-Loop Data Analytics (HILDA'16), San Francisco, CA, USA, 26 June 2016; ACM: New York, NY, USA, 2016. [CrossRef]
39. Robinson, I.; Webber, J.; Eifrem, E. *Graph Databases: New Opportunities for Connected Data*, 2nd ed.; O'Reilly Media: Sebastopol, CA, USA, 2015.
40. Fensel, D.; Şimşek, U.; Angele, K.; Huaman, E.; Kärle, E.; Panasiuk, O.; Toma, I.; Umbrich, J.; Wahler, A. Why We Need Knowledge Graphs: Applications. In *Knowledge Graphs: Methodology, Tools and Selected Use Cases*; Springer: Cham, Switzerland, 2020; pp. 95–112. [CrossRef]

41. Gomez-Perez, J.M.; Pan, J.Z.; Vetere, G.; Wu, H. Enterprise Knowledge Graph: An Introduction. In *Exploiting Linked Data and Knowledge Graphs in Large Organisations*; Springer: Cham, Switzerland, 2017; pp. 1–14. [CrossRef]
42. Negro, A. *Graph-Powered Machine Learning*; Manning Publications: New York, NY, USA, 2020.
43. Sowa, J.F. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*; Brooks/Cole: Pacific Grove, CA, USA, 2000.
44. Rodriguez, M.; Neubauer, P. Constructions from dots and lines. *Bull. Am. Soc. Inf. Sci. Technol.* **2010**, *36*, 35–41. [CrossRef]
45. Larriba-Pey, J.L.; Martínez-Bazán, N.; Domínguez-Sal, D. Introduction to Graph Databases. In Proceedings of the 10th International Reasoning Web Summer School—Reasoning on the Web in the Big Data Era (RW'14), Athens, Greece, 8–13 September 2014; Springer: Cham, Switzerland, 2014; pp. 171–194. [CrossRef]
46. Francis, N.; Green, A.; Guagliardo, P.; Libkin, L.; Lindaaker, T.; Marsault, V.; Plantikow, S.; Rydberg, M.; Selmer, P.; Taylor, A. Cypher: An Evolving Query Language for Property Graphs. In Proceedings of the 2018 International Conference on Management of Data (SIGMOD'18), Houston, TX, USA, 10–15 June 2018; ACM: New York, NY, USA, 2018; pp. 1433–1445. [CrossRef]
47. Francis, N.; Green, A.; Guagliardo, P.; Libkin, L.; Lindaaker, T.; Marsault, V.; Plantikow, S.; Rydberg, M.; Schuster, M.; Selmer, P.; et al. Formal Semantics of the Language Cypher. *arXiv* **2018**, arXiv:1802.09984.
48. Angles, R.; Gutierrez, C. An Introduction to Graph Data Management. In *Graph Data Management: Fundamental Issues and Recent Developments*; Springer: Cham, Switzerland, 2018; pp. 1–32. [CrossRef]
49. Hartig, O. Reconciliation of RDF* and Property Graphs. *arXiv* **2014**, arXiv:1409.3288.
50. Holzschuher, F.; Peinl, R. Performance of Graph Query Languages: Comparison of Cypher, Gremlin and Native Access in Neo4j. In Proceedings of the Joint EDBT/ICDT 2013 Workshops (EDBT'13), Genoa, Italy, 18–22 March 2013; ACM: New York, NY, USA, 2013; pp. 195–204. [CrossRef]
51. Green, A.; Guagliardo, P.; Libkin, L.; Lindaaker, T.; Marsault, V.; Plantikow, S.; Schuster, M.; Selmer, P.; Voigt, H. Updating Graph Databases with Cypher. *Proc. VLDB Endow.* **2019**, *12*, 2242–2254. [CrossRef]
52. Fowler, M. *Patterns of Enterprise Application Architecture*; Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, USA, 2002.
53. JupyterHub. 2021. Available online: https://jupyter.org/hub (accessed on 20 January 2021).
54. Kluyver, T.; Ragan-Kelley, B.; Pérez, F.; Granger, B.; Bussonnier, M.; Frederic, J.; Kelley, K.; Hamrick, J.; Grout, J.; Corlay, S.; et al. Jupyter Notebooks—A publishing format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*; IOS Press: Amsterdam, The Netherlands, 2016; pp. 87–90. [CrossRef]
55. Perkel, J.M. Why Jupyter is data scientists' computational notebook of choice. *Nature* **2018**, *563*, 145–146. [CrossRef]
56. JupyterHub Docker Spawner. 2021. Available online: https://github.com/jupyterhub/dockerspawner (accessed on 20 January 2021).
57. Jupyter Notebook Deep Learning Stack. 2021. Available online: https://hub.docker.com/r/jupyter/tensorflow-notebook (accessed on 20 January 2021).
58. Empowering App Development for Developers | Docker. 2021. Available online: https://www.docker.com/ (accessed on 19 January 2020).
59. Fielding, R.T. Architectural Styles and the Design of Network-Based Software Architectures. Ph.D. Thesis, University of California, Irvine, CA, USA, 2000.
60. Flask. 2020. Available online: https://palletsprojects.com/p/flask/ (accessed on 20 January 2021).
61. Flask-RESTful. 2020. Available online: https://flask-restful.readthedocs.io/ (accessed on 20 January 2021).
62. Celery—Distributed Task Queue. 2018. Available online: https://docs.celeryproject.org (accessed on 20 January 2021).
63. React—A JavaScript Library for Building User Interfaces. 2021. Available online: https://reactjs.org/ (accessed on 20 January 2021).
64. Why Did We Build React? 2013. Available online: https://reactjs.org/blog/2013/06/05/why-react.html (accessed on 20 January 2021).
65. Skoutas, D.; Simitsis, A.; Sellis, T. Ontology-Driven Conceptual Design of ETL Processes Using Graph Transformations. In *Journal on Data Semantics XIII*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 120–146. [CrossRef]
66. Chaudhuri, S.; Dayal, U. An Overview of Data Warehousing and OLAP Technology. *SIGMOD Rec.* **1997**, *26*, 65–74. [CrossRef]
67. Chen, C.; Yan, X.; Zhu, F.; Han, J.; Yu, P. Graph OLAP: A multi-dimensional framework for graph data analysis. *Knowl. Inf. Syst. Vol.* **2009**, *21*, 41–63. [CrossRef]
68. Zieglera, J.; Reimann, P.; Keller, F.; Mitschang, B. A Graph-based Approach to Manage CAE Data in a Data Lake. In *Procedia CIRP*; Elsevier: Amsterdam, The Netherlands, 2020; Volume 93, pp. 496–501. [CrossRef]
69. Beheshti, A.; Benatallah, B.; Nouri, R.; Tabebordbar, A. CoreKG: A Knowledge Lake Service. *Proc. VLDB Endow.* **2018**, *11*, 1942–1945. [CrossRef]
70. Drozdal, J.; Weisz, J.; Wang, D.; Dass, G.; Yao, B.; Zhao, C.; Muller, M.; Ju, L.; Su, H. Trust in AutoML: Exploring Information Needs for Establishing Trust in Automated Machine Learning Systems. In Proceedings of the 25th International Conference on Intelligent User Interfaces (IUI'20), Cagliari, Italy, 17–20 March 2020; ACM: New York, NY, USA, 2020. [CrossRef]