Web Semantics: Science, Services and Agents on the World Wide Web 71 (2021) 100664

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: www.elsevier.com/locate/websem

SemML: Facilitating development of ML models for condition monitoring with semantics

Baifan Zhou [a,b,c,*,1], Yulia Svetashova [a,d,*,1], Andre Gusmao [e], Ahmet Soylu [f,e,c], Gong Cheng [g], Ralf Mikut [b], Arild Waaler [c], Evgeny Kharlamov [h,c,**]

[a] Bosch Corporate Research Center, 71272 Renningen, Germany
[b] Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, 76344 Eggenstein-Leopoldshafen, Germany
[c] Department of Informatics, University of Oslo, 0316 Oslo, Norway
[d] Institute of Applied Informatics and Formal Description Methods, Karlsruhe Institute of Technology, 76133 Karlsruhe, Germany
[e] Department of Computer Science, Norwegian University of Science and Technology – NTNU, Teknologivegen 22, 2815, Gjøvik, Norway
[f] Department of Computer Science, OsloMet – Oslo Metropolitan University, Pilestredet 46, 0167 Oslo, Norway
[g] State Key Laboratory for Novel Software Technology, Nanjing University, 210023 Nanjing, China
[h] Bosch Center for Artificial Intelligence, 71272 Renningen, Germany

ARTICLE INFO

Article history:
Received 25 January 2021
Received in revised form 5 July 2021
Accepted 20 August 2021
Available online 1 October 2021

Keywords:
Condition monitoring
Ontologies
Templates
Data integration
Machine learning
Software architecture
Industry 4.0

ABSTRACT

Monitoring of the state, performance, quality of operations and other parameters of equipment and production processes, which is typically referred to as condition monitoring, is an important common practice in many industries including manufacturing, oil and gas, chemical and process industry. In the age of Industry 4.0, where the aim is a deep degree of production automation, unprecedented amounts of data are generated by equipment and processes, and this enables adoption of Machine Learning (ML) approaches for condition monitoring. Development of such ML models is challenging. On the one hand, it requires collaborative work of experts from different areas, including data scientists, engineers, process experts, and managers with asymmetric backgrounds. On the other hand, there is high variety and diversity of data relevant for condition monitoring. Both factors hampers ML modelling for condition monitoring. In this work, we address these challenges by empowering ML-based condition monitoring with semantic technologies. To this end we propose a software system SemML that allows to reuse and generalise ML pipelines for conditions monitoring by relying on semantics. In particular, SemML has several novel components and relies on ontologies and ontology templates for ML task negotiation and for data and ML feature annotation. SemML also allows to instantiate parametrised ML pipelines by semantic annotation of industrial data. With SemML, users do not need to dive into data and ML scripts when new datasets of a studied application scenario arrive. They only need to annotate data and then ML models will be constructed through the combination of semantic reasoning and ML modules. We demonstrate the benefits of SemML on a Bosch use-case of electric resistance welding with very promising results.

1. Introduction

Industry 4.0 [1] and technologies of the Internet of Things (IoT) [2] behind it lead to unprecedented growth of data generated in many industrial processes, such as manufacturing, oil and gas, chemical and process industries [3,4]. Indeed, modern machines and production systems are equipped with sensors that constantly collect and send data and with control units that monitor and process these data, coordinate machines and manufacturing environment and send messages, notifications, requests. Availability of these voluminous data has led to a large growth of interest in data analysis for a wide range of industrial applications [5–8], especially the use of Machine Learning (ML) approaches for *condition monitoring* for manufacturing processes, machines, oil, gas and chemical systems, and products by predicting system disturbance, machines' down-times or the quality of manufactured products [9]. Such approaches allow to analyse large amount of data and gain fruitful insights for condition monitoring.

* Corresponding authors.
** Department of Informatics, University of Oslo, 0316 Oslo, Norway.
E-mail addresses: baifanz@ifi.uio.no (B. Zhou), yulia.svetashova@de.bosch.com (Y. Svetashova), andrgusm@stud.ntnu.no (A. Gusmao), ahmet.soylu@oslomet.no (A. Soylu), gcheng@nju.edu.cn (G. Cheng), ralf.mikut@kit.edu (R. Mikut), arild@ifi.uio.no (A. Waaler), evgeny.kharlamov@de.bosch.com (E. Kharlamov).

[1] B. Zhou and Y. Svetashova contributed equally to this work as first authors.

https://doi.org/10.1016/j.websem.2021.100664

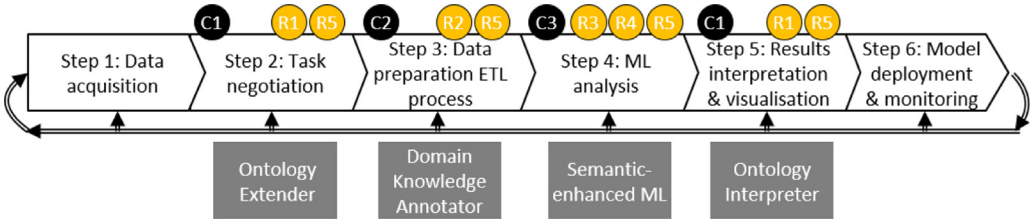

Fig. 1. Workflow of ML development for industrial data-driven condition monitoring with indications of *challenges* (C), *requirements* (R, see Section 1), and our semantic components. ETL stands for *Extract, Transform, Load* [10].

We summarise a typical workflow of ML-based condition monitoring in Fig. 1. The workflow is iterative and includes several steps: data collection (Step 1), task negotiation, to define feasible and economic tasks (Step 2), data preparation, to integrate data from different conditions and production environments (Step 3), ML analysis (Step 4), result interpretation and model selection (Step 5), and finally, model deployment in production (Step 6).

Development of such ML approaches is a complex and costly process where the following three challenges are of high importance for many companies, including Bosch, since they consume more than 80% of the overall time of development [11]. The first challenge (C1) is *communication*: Steps 2 and 5 of the workflow require collaborative work of experts from different areas, including data scientists, engineers, process experts, and managers that have asymmetric backgrounds, which makes communication time consuming and error-prone. The second challenge (C2) is *data integration*: Step 3 requires to integrate data from dozens of sources with highly manual modification. The third challenge (C3) is *generalisability* of ML models: each ML model developed in Step 4 is typically tailored to a specific dataset and one application scenario. Thus, reuse of this ML model for other data or scenarios requires a significant effort, while the reuse is highly desired, considering the wide spectrum of processes, equipment, and locations of industrial conglomerates. In Fig. 1 we annotated Steps 2–5 with the challenges as C1–C3 for clarity.

In this work we address the C1–C3 challenges by enhancing machine learning development for condition monitoring with semantic technologies. Note that semantics has recently gained a considerable attention in industry in a wide range of applications and automation tasks such as modelling of industrial assets [12] industrial analytics [13], integration [14–16] and querying [17–19] of production data, process monitoring [20] and equipment diagnostics [21], moreover, semantic technologies have been adopted or evaluated in a number of large high tech production companies such as Equinor [22], Siemens [23], Festo [24], and Bosch [25,26].

In particular, we developed an ontology-based software system, called SemML, that extends the conventional ML workflow with four semantic components that are depicted with grey boxes in Fig. 1. These components rely on ontologies, ontology templates, and reasoning. In particular, SemML exploits upper-level and concrete domain ontologies and the ML-ontology that captures machine learning tasks.The four semantic components of SemML are:

- **Ontology extender** that allows domain experts to describe domains in terms of an upper-level ontology by filling in templates. Data scientists then also use templates to annotate domain terms with task-related information. Then, they use the ontologies they jointly developed as a "lingua franca" for task negotiation.
- **Domain knowledge annotator** that enables data integration by annotating, mapping raw data to the terms in domain ontologies with ontology-to-data mappings.
- **ML knowledge reasoner** that uses automated reasoning to infer ML-relevant information from ontology-to-data mappings and creates the mappings between feature groups in ML ontologies and data for each raw data source.
- **ML interpreter** that facilitates uniform and explainable inspection of ML models and raw data.

Ontology extender and ML interpreter help us to address the communication challenge, domain knowledge annotator addresses the data integration challenge, and ML knowledge reasoner addresses the generalisability challenge.

SemML allows to store in a catalogue a set of pre-configured ML pipelines for condition monitoring that are described in terms of ontologies. Then, users can (re-)use these pipelines by deploying them on new datasets. This can be achieved by manually annotating raw data with domain terms and by selecting a suitable ML pipeline from the catalogue. Then, the system will automatically reason and construct ML models to analyse the data for the given quality monitoring task. SemML also allows users to extend the domain ontologies whenever needed, e.g., when existing ontologies do not contain domain terms required for data annotation. Thus, our ontology-based ML system SemML allows users to do ML based condition monitoring on a specific domain without an extensive knowledge of ML thanks to the semantic artefacts and ML pipelines offered by the system.

We demonstrate an application scenario of SemML with a Bosch use case of automatic welding during car manufacturing [27,28], which is the process of connecting two pieces of metal from car bodies together by passing high voltage electric current through them. In particular, we conducted a prototype deployment of our solution on Bosch data and conducted a user study with two experiments with 14 Bosch experts, including data scientists, measurement experts, and domain experts from two welding processes: resistance spot welding (RSW) and hot-staking (HS). For the deployment, we prepared data that was collected in process development phase at Bosch by a Finite-Element-Method (FEM) simulation model and anonymised, including 13952 welding operations (estimated to be relevant to 30 cars). The data include inputs of 235 single features and 20 time series. These data is relevant for two condition monitoring task: one task is to estimate the welding spot diameter, typically used to quantify the welding quality according to industrial standards [29,30], and the second task is to predict the quality of the next welding operation based on the quality of previous operations. For the user experiment, we developed a set of templates, domain ontologies, and welding process monitoring tasks. The users were first asked to create their domain ontologies using the ontology extender, and then map the variable names in raw data to the datatype properties of their created ontologies. After each task, they answered questionnaires to provide information on subjective satisfaction. The time and accuracy of these tasks and the scores of the questionnaires were recorded, analysed, and evaluated with promising results.

The paper is organised as follows: in Section 2 we describe the general problem of data-driven industrial condition monitoring,

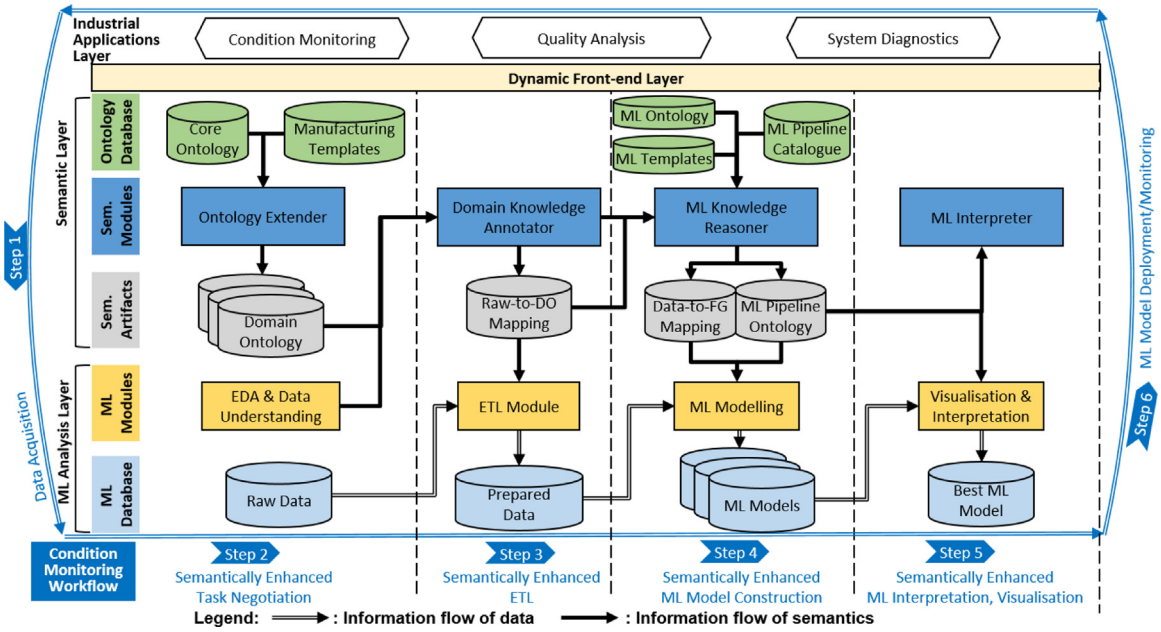

Fig. 2. An architectural overview of our semantically enhanced ML solution SemML for condition monitoring, where we overlay the welding quality monitoring workflow of Fig. 1 and the use-case requirements. EDA: exploratory data analysis, Sem.: semantics, Eng.: engineered.

elaborate the workflow of ML development and its challenges in each step in detail, and state the requirements of the system to enhance the workflow. In Section 3 we introduce our solution SemML, semantically-enhanced machine learning software system, its software architecture, working mechanism, semantic artefacts and system implementation. In Section 4 we introduce the Bosch use case of electric resistance welding. In Section 5 we demonstrate the application of SemML on the Bosch use case, which exemplifies the more detailed mechanism of SemML. In Section 6 we evaluate SemML in a user study: we first explain the experimental settings of the user study and then give its results and interpretation. In Section 7 we reviews some related work and in Section 8 we summarise our lessons learned, conclude the paper and discuss future research.

The material presented in this journal paper extends two of our previously presented conference papers [10,31]. First, we provide an extensive description of the ML development workflow (Section 2.2) which was only very shortly introduced in [10]. Second, we introduce a novel software architecture (Fig. 2) with significant improvement compared to [10]. Third, we add a new system implementation section with many details of software structures and graphic user interface. Fourth, we present three novel semantic artefacts, the Hot-staking ontology QMM-HS, ML templates, and the ML Pipeline ontologies. Fifth, we demonstrate a new mechanism of the semantic enhanced ML model construction (Fig. 10) of "Static Mode" that was not presented in [31]. Sixth, we elaborate on extensive detailed information of the software components, use cases, ML pipelines, etc. Also note that some of the aspects of SemML were also discussed in our demo, poster, or industry track presentations [27,32,33]. A comparison of ML methods with extensive evaluation is presented in [34].

2. Data-driven industrial condition monitoring

We now introduce data-driven condition monitoring, its workflow and requirements.

2.1. Condition monitoring

In industry condition monitoring refers to techniques and methods for monitoring of some parameters of condition in production machinery in order to identify a significant change which is indicative of a developing fault. [35] In particular, two types of condition monitoring are typically considered: (1) *Machinery monitoring* [9] is about monitoring of how healthy is the current state of an operating system or equipment; (2) *Process quality monitoring* is about monitoring of how well a production process go and its product quality.

Examples of industrial scenarios for condition monitoring are numerous. For process monitoring they include quality monitoring of individual operations in manufacturing processes such as welding that connects pieces of metal together. The quality indicators of welding systems provided by Bosch are monitored by analytic methods [31] or destructive methods [29,30]. For machinery monitoring examples include data-driven monitoring of trains [21] and turbines [15] in Siemens. In the Oil and Gas industry [36], examples includes equipment and process monitoring in off-shore platforms and oil reservoirs at Equinor [14]. Another example is the detection and processing of disturbances in chemical or process industry for root-cause-analysis at ABB and INEOS [8].

As the technologies of Internet of Things [2] and Industry 4.0 [1] develop, condition monitoring often involves a large amount of data. The core of condition monitoring is often to estimate or forecast some categorical or numerical indicators with intelligent data-driven methods like machine learning. A series of steps are dedicated to this end: data need to be collected, concrete tasks in different domains need to be determined, the data need to prepared, analysed by ML modelling, the results of analysis need to be interpreted, and at the end the ML models can be deployed in industry.

2.2. Workflow of ML development

We first describe a generic workflow of the machine learning development for industrial condition monitoring without enhancement of semantic technologies. It is adapted from the workflow proposed by Fayyad [37] and Mikut [38] (Fig. 1).

Step 1, before the question is defined and data analysis is requested, data are collected by some system monitoring software, including e.g. installed sensors configured by domain experts, or manually by the measurement experts. Data scientists will

receive an initial historical dataset, provided as given. For each welding operation, the collected data can contain various data formats, e.g. sensor measurements along time, (referred to as time series in data science), single valued quantities (referred to as single features in data science) describing the machine conditions like wearing and maintenance, the material and geometrical properties of system components or products, etc. In a later stage, data scientists can also request to collect further data after some findings are revealed, indicated by the arrows pointing backwards from the later stages in Fig. 1. Involved stakeholders in this stage are measurement experts for design of measurement settings, process experts for design of experiments, and data scientists for specifying data collection request. The challenges here is the effective and economic design of experiments and measurement settings, and specification of data collection request.

Step 2, the process experts introduce the welding process and question of quality monitoring. The measurement experts present the initial dataset and relevant information. The managers and data scientists both perform an initial question evaluation to check if the question suits the strategic interest of the project, whether it is feasible to solve the question with state-of-the-art of machine learning, what adaptation is necessary, and request further needed resources. Involved stakeholders are all parties. The activities in this stage include understanding of machine learning for the managers, process experts, and measurement experts so that they can propose economically solvable questions, and understanding of the data, process, and question for the data scientists, so that they acquire necessary domain knowledge. To enable the understanding, all parties need to communicate very closely; however, a challenge lies here. Practice has shown that the *communication and mutual understanding* between these experts can be onerous and error-prone. Despite of using the same vocabulary, the exact meanings conveyed by these stakeholders can often be discrepant. The reason is the information asymmetry caused by their varying disciplinary backgrounds. A "lingua franca" is needed to standardise the definitions of concepts in the domain and to organise knowledge in order to facilitate communication between the stakeholders.

Step 3, data collected under different conditions, locations, customers, experiment designs, measurement settings, software systems, recorded in different formats, measured with different equipment need to be prepared into some uniform data format before they can be processed or analysed. The data may have different names for the same variables, or have some variables missing in one source but present in another source, or measured with different sampling rate, etc. Adding to the production data, data collected from laboratory and simulation for process development can have more discrepancies. Extra sensors are installed in the laboratory, and the simulation data are generated with different mechanisms, despite that they represent the same welding process. The various data sources bring difficulty to the practice to analyse them. Even data collected from the same laboratory or simulation, can have different versions with different design of experiments in different stages of process development. It would be extremely costly and time-consuming to develop different data analysis applications for each data source or format. A better solution is to systematically organise these data and integrate them in a uniform way [39]. The challenges here are to prepare these data, by renaming of the variables, unifying sampling rate of the time series, synchronising the starting time stamps of time series, etc, or for short, to integrate these data in an uniform format and store them in one data base so that they can be further processed by machine learning algorithms. The data scientists or data managers need to stay in close contact with process experts and measurement experts, since without a proper understanding of the data and process, it is impossible to integrate data meaningfully and facilitate the later data analysis. Involved stakeholders are data scientists, process experts and measurement experts.

Step 4, ML analysis includes adequate data preprocessing and its subsequent machine learning modelling. Given a input set $\mathcal{X}$ which contains input features of data samples $\{x|x \in \mathcal{X}\}$ and its output set $\mathcal{Q}$, which contains quality indicators of the corresponding data samples $\{q|q \in \mathcal{Q}\}$ the goal of machine learning [40] is to find a hypothesis of statistic estimation $h|P : \mathcal{X} \rightarrow \hat{\mathcal{Q}}$ (where $\hat{\mathcal{Q}}$ is the estimation of Q), that is to obtain P, the set of parameters of h, by calculating

$$P = \arg\min_{P} L_{\mathcal{D},f}(h|P) \tag{1}$$

where the loss function $L_{\mathcal{D},f}(h|P)$ is defined as

$$L_{\mathcal{D},f}(h|P) = \mathcal{D}(\{x : h(x|P) \neq q\}) \tag{2}$$

which is namely to minimise the likelihood of observing $h(x) \neq q$, where $\mathcal{D}(x)$ gives the likelihood of observing x.

How to achieve this hypothesis $h|P$ with accurate prediction (minimum loss) is the very question machine learning study attempts to answer. Industrial applications often require the data analysis to be not only accurate, but also interpretable and can provide insights for process experts. According to the literature, machine learning pipelines in manufacturing can be largely divided into two schools [41]. Among which, an important school is feature engineering, which is to generate new features by changing the representation of the data [42] to improve modelling performance, and render the machine learning more transparent by make feature evaluation and interpretation possible. However, feature engineering is laborious and time-consuming [42], because different domains have distinctive features, entailing different feature engineering strategies, which is a main reason that the other school, featured by automatic feature learning (currently mainly deep learning) is so hailed. It is highly desired that the development of machine learning approaches for other similar manufacturing processes can learn something from conducted projects and require less effort. The challenges in this stage are therefore to develop understandable machine learning approaches, incorporate a-prior domain knowledge provided by the process experts, and make the developed approaches generalisable for other similar manufacturing processes. Involved stakeholder in this stage is mainly data scientists.

Step 5, the data scientists need to present the results, and discuss intensively with other stakeholders to interpret the results, the machine learning models meaningfully, draw appropriate conclusions. The managers need to make thoughtful decisions. For example, is the developed approach sufficient accurate and responsive? Can it be implemented in the production hardware? What improvement is necessary? To achieve the improvement are more data required so that a new iteration of data collection should be performed?

Step 6, managers then have to choose the best model to be deployed in industry. These models will be under constant monitoring for addressing concept drift issues.

The workflow is intrinsically iterative. At any stage this workflow can direct backwards to any of the previous stages.

2.3. Use case requirements

Summing up, the time and effort required for ML development is heavily affected by *(C1)* the necessity of multiple iterations of communication by different stakeholders, *(C2)* the complexity of the data integration process, and *(C3)* generalisability of the developed ML models to similar processes and datasets. In order to enhance the ML workflow in a way that it addresses C1–C3 we derived the following five system requirements:

- *R1: Uniform communication model for various stakeholders*: The system should rely on a common vocabulary, with unambiguously defined relations between the terms. This vocabulary should be machine-readable and minimally controversial.
- *R2: Uniform data format and ML vocabulary*: the results of the ETL process are the input to ML modelling. Thus, the system should offer a uniform format for the data storage and a uniform naming of variables.
- *R3: Mechanism for generalising ML models*: the system should offer a mechanism for machine learning methods developed on one dataset to be reused or generalised to other datasets and manufacturing processes.
- *R4: Data enrichment mechanism*: the system should enable the enrichment of data with some task-specific information so that the integrated data can be linked to the generalisable machine learning approaches.
- *R5: Flexibility, extensibility, maintainability*: the system and its functionalities should enable accommodation of new data sources and ML tasks.

Note that the requirements R1 and R5 address the challenge C1, then R2 and R5 address C2, and R3–R5 address C3; we depict it in Fig. 1 with yellow circles.

3. Semantic Solution for Data-Driven Industrial Condition Monitoring

In this section, we present our semantically enhanced machine learning system, SemML, its architecture, mechanism and implementation. SemML enhances Step 2 to Step 5 of the workflow in Fig. 1. The section is organised as follows: in Section 3.1 we present an overview of the system architecture, in Section 3.2 we lay out stepwise details of SemML components following the steps of the workflow in Fig. 1, in Section 3.3 we zoom into details of SemML implementation, and finally in Section 3.4 we present the main semantic artefacts of SemML.

3.1. SemML system architecture

SemML has a modular and multilayered architecture illustrated in Fig. 2. In order to simplify for the reader the understanding of how SemML works, we overlay the architecture with the workflow from Fig. 1 where the steps are indicated with blue arrows. SemML has four layers: *Industry Applications Layer* where the condition monitoring, diagnostics, and quality analysis happen, *Dynamic Front-end Layer* that presents GUI to the users to conveniently use the software system without diving deep into the semantic or ML knowledge, *Semantic Layer* that contains the ontology database with pre-designed ontologies and templates libraries, semantic modules that allow users to access the ontology database and create their semantic artefacts for the specific tasks the users want to solve, and semantic artefacts created by the users. *ML Analysis Layer* that contains machine learning modules enhanced with our semantic modules and the ML database which stores the data collected from industry and the ML models generated by the software system.

The arrows in Fig. 2 indicate the data flow from the raw data sources generated by the industry application of condition monitoring, through the machine learning analysis steps, and back to the top layer where the developed quality models are deployed and monitored.

We now discuss the mechanism of SemML and work the reader through the workflow with semantic enhancement.

3.2. Mechanism of the semantic enhanced ML

Step 2: Semantically Enhanced Task Negotiation. Once the raw data are acquired, data scientists and process experts align their backgrounds and specify the task of quality analysis. To this end, we developed the module of semantic *Ontology Extender*, which can be combined with ML module of *Exploratory Data Analysis* (EDA) for process and data understanding. Its graphical user interface (Fig. 4.1) allows experts to describe their domain in terms of an upper-level ontology that encodes general knowledge of manufacturing (or oil and gas, chemical and process industries, etc.), named as *Core Ontology* in this work, by filling in *Ontology Templates*. The users thus create domain ontologies that reflect the specificity of the raw data and a manufacturing process. For example, based on the Core Ontology and templates, users create their domain ontologies for different welding processes, such as *rsw* for resistance spot welding and *hs* for hot-staking. Although domain ontologies are created by the users, they are ensured to be of very good quality since they are built strictly based on the Core Ontology and Ontology Templates. Templates are also used by data scientists to annotate domain terms with task-related information. Thus, Ontology Extender, as well as the Core ontology and ontology Templates, addresses the R1 and R5 requirements: the Core ontology serves as a common communication model and the templates make the system flexible and extensible to new data sources.

Step 3: Semantically Enhanced ETL. Our *Domain Knowledge Annotator* enables data integration via the mapping of the raw data to the terms in the domain ontologies. For mappings, we introduce a compact graphical user interface (Fig. 4.2) with browsing functionalities of the ontology and it is linked to the Ontology Extender. In case when a required term is missing, the user can switch to the Ontology Extender, and the newly introduced term immediately becomes available for use. A domain knowledge mapping (*Raw-to-DO Mapping*, where DO stands for Domain Ontology) is generated by the user activities. It is then used by the *Extract-Transform-Load* (ETL) module to prepare the data for machine learning, as shown in the semantic layer of Fig. 2. For example, the features *Strom*, *CurrentAmp*, and *CurrentCurve* from different data sources of resistance spot welding (RSW) are all mapped to the term *rsw:Current*, and thus all will be renamed to *rsw:Current*. The raw data from various formats are transformed to prepared data with uniform agreed format, and the different feature names of different data sources in the raw data are changed to unified feature names. Thus, the Domain Knowledge Annotator addresses the R2 and R5 requirements.

Step 4: Semantically Enhanced ML Model Construction. The data prepared after the semantically enhanced ETL go through the *ML Knowledge Reasoner* module. It has a graphic user interface (Fig. 4.3) to allow the user to select a ML pipeline from the ML pipeline catalogue, which stores several ML pipelines developed beforehand by ML experts and encoded in ML pipeline ontologies. Then, the ML Knowledge Reasoner relies on the ML Ontology (referred to as *ml*) and an ontology reasoner to infer machine learning-relevant information from the Raw-to-DO mappings, that is to infer the ML feature groups (FG) for each source from the Raw-to-DO mapping. This step is fully automated. For example, sensor signals measured along time such as *rsw:Current*, *rsw:Voltage* and *rsw:Resistance* are categorised as the ML feature group *ml:FG-ProcessCurves*, which is of feature type *ml:TimeSeries* and can be processed by specific algorithms. It also exams available feature groups in the dataset, and adjusts the selected ML pipeline to the dataset, and results a ML pipeline ontology tailored to the dataset. By doing so, the raw data are fully docked with the ML pipeline that the user selects. It creates

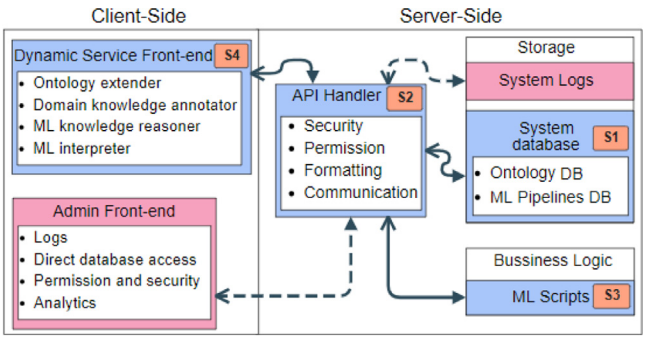

Fig. 3. Low-level system structure design. Core structures have light blue colour and provide the main functionalities . Support structures provide management functionalities only and are presented with pink colour. Solid double-headed arrows represent the communication of main functionalities and dashed double-headed arrows the communication of management functionalities.

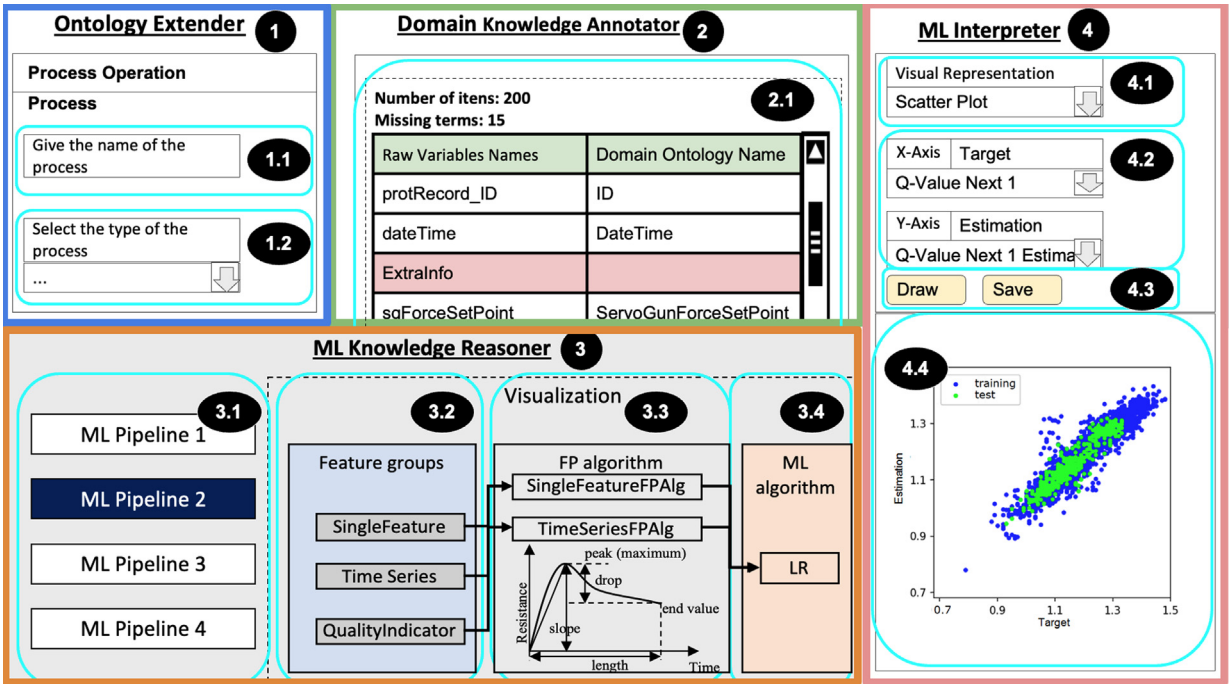

Fig. 4. Graphical user interfaces for (1.1–1.2) Ontology Extender, (2.1–2.2) Domain Knowledge Annotator, (3.1–3.4) ML Knowledge Reasoner and (4.1–4.4) ML Interpreter. A more detailed view of Ontology Extender and Domain Knowledge Annotator see Fig. 14.

the *Data-to-FG Mapping* for each raw data source. The resulting two kinds of mappings store different relationships. Indeed, consider for example a sensor measurement feature named as "CurrentAmp" that contains a series of observations of electric current values with time stamps. This feature will be mapped to the domain term "operationCurveCurrentValue" with a Raw-to-DO mapping and to the ML term "FG-ProcessCurves" with an Data-to-FG mapping. The latter indicates that this column will be treated as process curves, which of feature type time-series in machine learning. Data-to-FG mappings enable the uniform handling of the prepared data by ML algorithms in the *ML Modelling* module. This module performs various transformations of data categorised as feature groups and construct several machine learning models trained on the data. Our ML Knowledge Reasoner addresses the requirements R3–R5.

Step 5: Semantically Enhanced ML Interpretation and Visualisation. In order to conduct ML interpretation, data scientists discuss the ML models with other stake-holders through the *Visualisation & Interpretation Module*. Our *ML Interpreter* module (Fig. 4.4) facilitates a uniform and explainable inspection of ML models and raw data using ontologies, and thus, addresses the requirements R1 and R5. After the inspection, a selected ML model, and insights provided by ML analysis are deployed in the industrial applications layer.

3.3. System implementation

In the low-level design of SemML, it was decided to divide the system into four core structures and two support structures represented in Fig. 3. The core structures were developed in order to provide the functionalities corresponding to the semantic components described in Section 1. The support structures provide functionalities required for the management of the system. The core structures are described as following:

- *S1: System database*: This structure is responsible for storing ontologies, their mappings to the raw variables, and ML pipelines. The structure is divided into two sub-databases: ontology database and ML database. The ontology database contains all semantic artefacts. The ML database contains the raw datasets and prepared datasets to be analysed and the generated ML models.

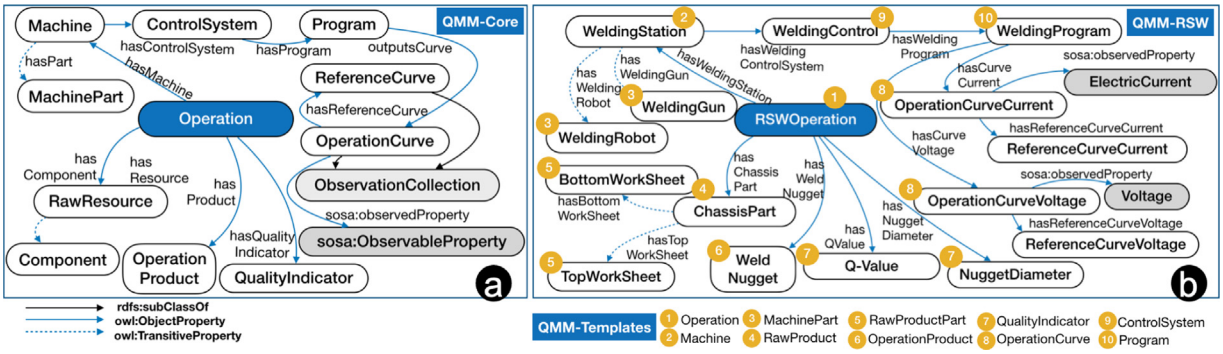

Fig. 5. Schematic illustration `QMM-Core`, `QMM-RSW` and Templates where prefixes such as `qmm-core` are omitted [10].

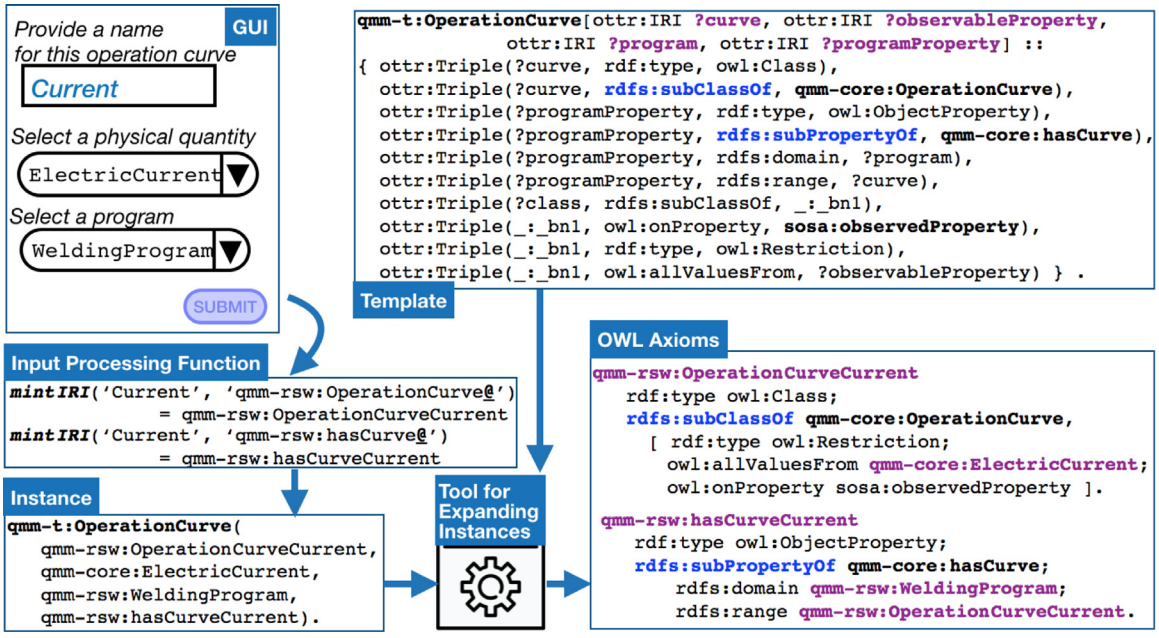

Fig. 6. Example to illustrate the mechanism of ontology extension based on templates by the users. In the GUI (top left) the user fills the variables of the template, *qmm-t:OperationCurve*. The user input is processed by the Input Process Function and creates a Template Instance. The Tool for Expanding Instances relies on the Template Instance and the Template, creates the class *qmm-rsw:OperationCurveCurrent*, and serialises the class in OWL axioms [10].

- *S2: API Handler*: The API handler connects all other three structures. It is designed as a REST API handling and structuring the requests and responses. This structure also manages the permissions and security of the system.
- *S3: ML components*: This structure contains machine learning scripts and is responsible for storing and executing machine learning algorithms on request over the data provided via the API Handler. The results and processed data are then sent back as requested by the API handler.
- *S4: Dynamic Front-end*: This structure enables users to interact with the system functionalities and data. It contains the user interfaces required for accessing the system components and data used during ML processes. The front-end structure covers the following components: Ontology Extender, Domain Knowledge Annotator, ML Knowledge Reasoner, and ML Interpreter.

The four semantic components described earlier play a critical role in the overall user experience, since they require involvement of the end users in the critical parts of the ML processes. In this respect, these components require robust graphical user interfaces. We designed GUIs for these components as depicted in Fig. 4. A Web based implementation, including Web technologies such as HTML, JavaScript, and CSS, is decided for the user interfaces in order to provide a platform independent solution. A widget-based approach is preferred for connecting individual sub-interfaces providing a smooth flow and well-connected user experience [43]. The user interfaces are described as follows:

- *GUI 1: Ontology Extender*: The Ontology Extender is capable of achieving the designed tasks assigned for it in Section 1 and is illustrated in Fig. 4.1. It is possible to assign a customised name to the process via text box in 1.1. With the name assigned to the process the user can select the type of the process in the drop-down list in 1.2.
- *GUI 2: Domain Knowledge Annotator*: This GUI component presented in Fig. 4.2 allows the user to visualise the automatically assigned domain ontology names to the raw variables in 2.1.
- *GUI 3: ML Knowledge Reasoner*: The GUI presented in Fig. 4.3 shows a selection of predefined machine learning pipelines that can be used for the input data, visualised in 3.1. With the feature groups by the ML Knowledge Reasoner, the data are automatically annotated with these feature groups and are used in the selected machine learning pipeline. A visualisation of these assigned feature groups to be used in

the pipeline can be seen in 3.2. The feature processing algorithms (FP algorithms) and the subsequent ML algorithm are illustrated in 3.3 and 3.4.

- *GUI 4: ML Interpreter*: The interface represented in Fig. 4.4 allows users to visualise and inspect the machine learning models and the prediction results. The desired visual representation can be selected among predefined representations as depicted in 4.1, with the labels of X-Axis and Y-Axis configured in 4.2. The resulting visualisation can be drawn in the browser canvas or saved locally by clicking the buttons in 4.3. An example of the drawn visualisation is depicted in 4.4.

In Fig. 6 we exemplify our template instantiation process with one template qmm-t:OperationCurve. Templates guarantee uniformity of the updates and the consistency of the updated ontology, as well as the relative simplicity of the ontology extension process.

3.4. Semantic artefacts of SemML

Semantic layer of SemML customises the system to different domains and usage scenarios through automated reasoning. The core of the solution comprises the following seven semantic artefacts. We first list these artefacts and then explain them in detail. Since these artefacts were developed for quality monitoring in manufacturing, they have the prefix *QMM*.

1. ***QMM-Core Ontology***, the upper-level ontology for quality monitoring in manufacturing, encoding general knowledge of manufacturing;
2. ***QMM-Templates***, the library of ontology templates for domain knowledge encoding;
3. ***Domain Ontologies***, specific ontologies for different domains, constructed by the users based on ***QMM-Core*** and ***QMM-Templates***;
4. ***QMM-ML Ontology***, the task ontology for machine learning that powers the ML components of the system;
5. ***ML-Templates***, the library of ontology templates for ML knowledge encoding;
6. ***ML Pipeline Catalogue***, which stores a set of ML solutions developed by ML experts beforehand and encoded in ML pipeline ontology;
7. ***ML Pipeline Ontologies***, which contain concrete ML solutions for specific datasets. The selected ML pipeline ontology from the ML Pipeline Catalogue by the user will be adjusted to the dataset by SemML.

We now describe these semantic artefacts and show how templates enable the construction of domain ontologies and the reusability of ML pipeline descriptions.

QMM-Core Ontology has been developed through a series of workshops, taking inputs from various Bosch experts of engineering and machine learning. It reflects the consensus terminology for a common base of discussion. QMM-Core Ontology is an OWL 2 ontology and can be expressed in the Description Logics $\mathcal{S}(\mathcal{D})$. With its 1170 axioms, which define 95 classes, 70 object properties and 122 datatype properties, it models the processes of discrete manufacturing with an emphasis on quality analysis. The left part of Fig. 5 displays the main classes and relations between them.

QMM-Core takes an operation-centred perspective: all classes describing the context of the operation, equipment and product are linked to the class *qmm-core:Operation*. This orientation naturally follows from the analytical task of quality prediction described in Section 2.1. In particular, a *qmm-core:Operation* is performed by a *qmm-core:Machine* on a *qmm-core:RawProduct*. It results in a *qmm-core:OperationProduct*. Sensor observations are stored as *qmm-core:OperationCurves* and represent series of observation results with their corresponding timestamps. This class is our lightweight adaptation of *ssn-ext:ObservationCollection* from the proposed extensions to the Semantic Sensor Network Ontology [44]. We thus align QMM-Core with the established way to model and query sensor observations — the SOSA/SSN ontology [45].

QMM-Core presents the detailed modelling of the manufacturing equipment and the context of the operation through domain and range axioms of the object properties. The important group of object properties encodes part-whole relationships between classes: e.g. *qmm-core:hasPart* links *qmm-core:Machine* and *qmm-core:MachinePart* as well as two or more *qmm-core:MachineParts*. This allows to describe complex machines and product assemblies. Meronymic properties are defined as transitive. We make use of data properties to encode the characteristics of entities through the same domain-range axiom mechanism.

Classes and properties of the QMM-Core Ontology define the key modelling patterns for the domain ontologies of different manufacturing processes. Patterns capture repetitive structures in the linked classes and their associated properties and can be instantiated via the ontology templates.

QMM-Templates. Ontology templates can be seen as parametrised ontologies. Users of templates create ontology elements by providing values to these parameters. The values can be named classes, object and data properties, individuals, and plain literals. A template is instantiated by the replacement of its parameters by the provided values. Instances of templates are expanded to OWL 2 axioms. To record templates we rely on the Reasonable Ontology Templates (OTTR) framework [46] and we use the template processor tool Lutra.[2]

For our use cases, we created 30 templates that rely on the classes and relationships of the QMM-Core ontology. In the following, we refer to this template library as QMM-Templates.

Domain Ontologies are created by the users based on the *QMM-Core* by filling the templates. They share the same knowledge patterns in the *QMM-Core* and the templates. For example, in Fig. 5.b the *QMM-RSW* mimics the knowledge pattern in the *QMM-Core*, but encodes the particularities of resistance spot welding. In particular, a *qmm-rsw:RSWOperation* is performed by a *qmm-rsw:WeldingStation* on a *qmm-rsw:ChassisPart*. It results in a *qmm-rsw:WeldNugget*. Sensor observations are *qmm-rsw:OperationCurveCurrent* and *qmm-rsw:OperationCurveVoltage*, which are sub-classes of *qmm-core:OperationCurve*. With this example, it can be clearly seen that the common knowledge patterns can be emphasised across domain ontologies.

All classes created by the same ontology template share the corresponding super-classes from the QMM-Core Ontology and a set of related properties, linked to the QMM-Core Ontology by the *rdfs:subPropertyOf* axioms. The users of SemML annotate datasets with domain classes and properties, and the mentioned linkage allows to automated the handling of these elements at a higher abstraction level, namely at the level of groups of features, in the ML Modelling modules of SemML.

QMM-ML Ontology has classes to categorise features as *qmm-ml:FeatureGroups*: time series, categorical features, identifiers, etc. It also encodes various preprocessing, feature engineering, and ML algorithms. QMM-ML is partially depicted in Fig. 7. It contains 62 classes, 4 object properties, 2 datatype properties as well as 210 axioms and 122 annotation assertions; it can be expressed using $\mathcal{ALH}(\mathcal{D})$ Description Logics.

[2] https://gitlab.com/ottr/lutra/lutra.

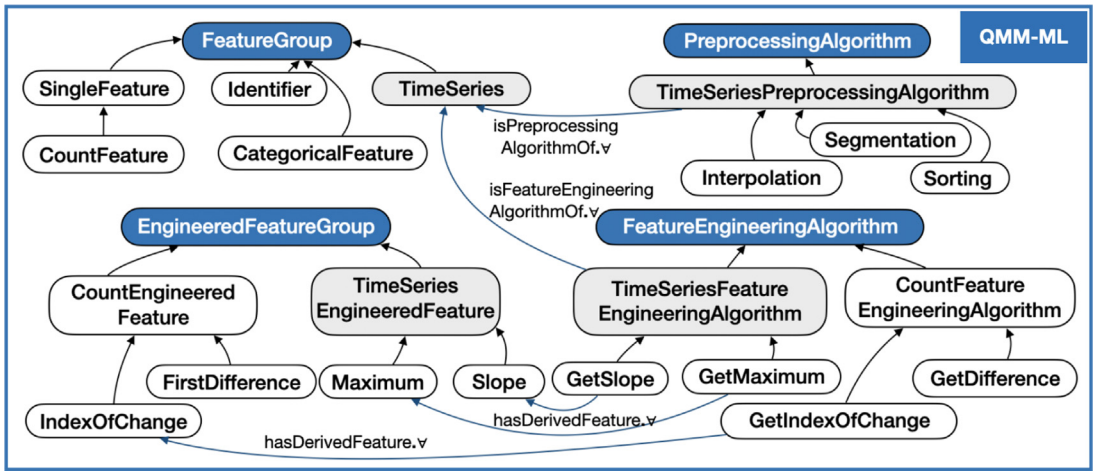

Fig. 7. A fragment of the QMM–ML ontology, which gives examples of feature groups, preprocessing and feature engineering algorithms and engineered features [10].

QMM-ML Ontology is used to enhance the dataset described in the Domain Knowledge Annotator with the ML-relevant information on the feature level. This is done via reasoning and the reasoning results are stored as Data-to-FG mappings store *qmm-ml:FeatureGroups* for all columns in the prepared data.

The ML Modelling module of SemML, in turn, has generic operations and algorithms with the behaviour specified on the level of *qmm-ml:FeatureGroups* of QMM-ML. Thus, the ML Knowledge Reasoner can retrieve the pre-processing and feature engineering algorithms for each group of features and automatically derive the types and the names of the engineered features. To this end, it relies on the corresponding class definitions in the *QMM-ML*. For instance, the pre-processing algorithm for time series is defined as follows:[3]

(1) Class: *qmm-ml:TimeSeriesPreprocessingAlgorithm*

(2) SubClassOf: *qmm-ml:isPreprocessingAlgorithmOf* only *qmm-ml:TimeSeries*

The ML Modelling module contains the implementation for each of the algorithm's subclasses: *qmm-ml:Interpolation*, *qmm-ml:Segmentation* and *qmm-ml:Sorting*.

The feature engineering algorithm for time series is defined analogously:

(3) Class: *qmm-ml:TimeSeriesFeatureEngineerAlgorithm*

(4) SubClassOf: *qmm-ml:isFeatureEngineerAlgorithmOf* only *qmm-ml:TimeSeries*

Its subclasses, in turn, are related to the corresponding engineered features, and the ML Modelling module will have the implementation for all of them. For example, based on the definition:

(5) Class: *qmm-ml:GetMaximum*

(6) SubClassOf: *qmm-ml:TimeSeriesFeatureEngineer-Algorithm*

(7) SubClassOf: *qmm-ml:hasDerivedFeature* only *qmm-ml:Maximum*

The ML Modelling module will apply the implemented GetMaximum algorithm to all time series features and generate new features with the token "Maximum" in their name for all of them.

In ML terms, the way how our semantically enhanced ML Modelling module works is: $h : \mathcal{X} \xrightarrow{M} \{\{FG_1\} \cdots \{FG_N\}\} \xrightarrow{\text{QMM–ML}} \{\{FPG_1\} \cdots \{FPG_K\}\} \rightarrow \hat{\mathcal{QI}}$, where h is a hypothesis that maps raw input features $\mathcal{X}$ into an estimation $\hat{\mathcal{QI}}$ of a welding quality indicator $\mathcal{QI}$. This mapping has two intermediate steps: (1) using Data-to-FG Mapping M it fetches a set of standardised Feature Groups *FG*s and (2) using QMM–ML it turns them into a set of Feature Processed Groups *FPG*s (*EngineeredFeatureGroup* is a subclass of FPG). This makes the developed ML approaches easily extendable to similar tasks and datasets. Moreover, this enables non-ML-experts to better understand the ML approaches, and even to modify the ML approaches with minimal training of ML expertise. Note that a classical ML Modelling module starts with $\mathcal{X}$ and may develop different ad hoc feature processing strategies for different tasks and data sources to estimate $\hat{\mathcal{QI}}$, or schematically: $h : \mathcal{X} \rightarrow \hat{\mathcal{QI}}$.

ML-Templates. This library essentially contains three groups of templates: (1) templates that describe the meta-information, e.g. relationship between datasets and entries; (2) templates that connect the annotation input by the users as domain terms to the *QMM-Core* and to the *QMM-ML*; (3) templates that can be used to construct ML pipeline ontologies, e.g. the structure of ML pipelines from starting layer (prepared data layer), to feature processing layer (data preprocessing), to ML modelling layer, and to the end layer (ML models layer), the general pattern of input-algorithm-output.

ML Pipeline Catalogue. ML solutions are developed beforehand by ML experts and encoded as ontologies in the catalogue. These ML solutions are developed in a general manner, that is to say, for example, their performance is relatively insensitive to hyperparameter changes, or they can cover a quite broad range of application scenarios. Since ML solutions often consist of a series of steps, taking prepared data as input, with many modules of data preprocessing and ML modelling, and outputting ML models at the end, they are referred to as ML pipelines in this work.

ML Pipeline Ontology. An ML Pipeline Ontology is an executable description of a concrete ML pipeline configuration. It adopts a layer-wise structure, which always starts from the *qmm-ml:PreparedDataLayer*, goes a series of *qmm-ml:FeatureProcessingLayer*, ends *qmm-ml:MLModellingLayer*. These are the three types of layers. The layers are connected with the object property *qmm-ml:hasNextLayer*. Each *qmm-ml:FeatureProcessingLayer* or *qmm-ml:MLModellingLayer* has a structure of *qmm-ml:Input*, *qmm-ml:Algorithm*, and *qmm-ml:Output*. A piece of the ML pipeline ontology looks like this:

(8) Individual: *qmm-ml-i:p1*

[3] We use the Manchester Syntax of OWL 2 [47], where classes and properties have prefixes *qmm-rsw:*, *qmm-core:*, *qmm-ml:* that indicate the ontologies they belong to.

(9) Types: *qmm-ml:Pipeline*

(10) Facts: *qmm-ml:hasNextLayer qmm-ml-i:l1*

(11) Individual: *qmm-ml-i:l1*

(12) Types: *qmm-ml:FeatureProcessingLayer*

(13) Facts: *qmm-ml:hasNextLayer qmm-ml-i:l2, :hasInputOutputCombination io1, io2*

(14) Individual: *qmm-ml-i:io1*

(15) Types: *qmm-ml:InputOutputCombination*

(16) Facts: *qmm-ml:hasInput :FG-SingleFeature, qmm-ml:hasAlgorithm qmm-ml:Maintain, qmm-ml:hasOutput :FG-SingleFeatureMaintained*

(17) Individual: *qmm-ml-i:io2*

(18) Types: *qmm-ml:InputOutputCombination*

(19) Facts: *qmm-ml:hasInput :FG-ProcessCurve, qmm-ml:hasAlgorithm qmm-ml:GetStats, qmm-ml:hasOutput qmm-ml:FG-TSStats*

We show the ML Pipeline ontology for the simplest pipeline schematically in Fig. 9. Note that all ML Pipeline ontologies are built by using ML-Templates: *qmm-t:Pipeline*, *qmm-t:Layer*, *qmm-t:Input-Output*, and *qmm-t:Target*. The latter template allows to mark a variable in a dataset as a predicted variable for ML.

4. Bosch use case: Quality monitoring in electric resistance welding

We now discuss the Bosch welding process quality monitoring use case, the collected data and problem definition.

4.1. Bosch welding process quality monitoring

Bosch is one of the global manufacturing leaders in the automotive industry. Welding is heavily used in industry for numerous applications including car production. Indeed, a typical car body can contain up to 6000 welding spots [48] where pieces of metal are connected. Bosch welding solutions include welding equipment (Fig. 8 for RSW) software, service, development support, etc. These solutions are used in Bosch plants and many customers worldwide, e.g. Daimler, BMW, Volkswagen, Audi, Ford. Enabled by the abundant data and computing resources behind the IoT technologies, Bosch is developing ML methods to predict the welding quality of next spots, before the actual welding happens. In the workflow illustrated in Fig. 8, the ML solutions should be able to predict the quality of the 6th welding spot, based on data of previous welded spots, including sensor measurements, welding configurations, past spot quality, etc. This allows to take necessary measures beforehand, like automatic adjustment of welding parameters, to improve the expected welding quality and avoid potential quality failure.

4.2. Bosch welding data

The welding quality is quantified by a quality indicator: Q-Value. It is developed empirically by Bosch Rexroth with long-time experience and engineering know-how. The optimal value of Q-Value is 1, indicating perfect quality. A Q-Value larger than 1 indicates too much energy is spend on the welding spot, while Q-Value smaller than 1 often means quality deficiency. Q-Values are typically computed on datasets collected from production lines. The data we collected for the use case consist of:

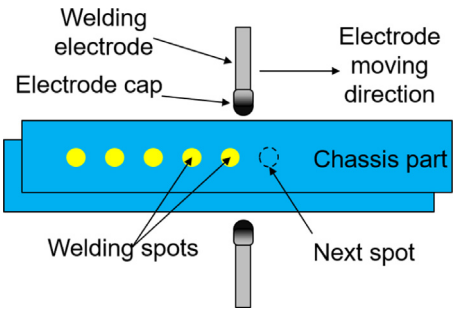

Fig. 8. Resistance Spot Welding (RSW) [31].

- four types of automatically generated *protocols* that contain descriptive information of the actual welding processes:
  - *main protocol*: with data recorded by the welding software systems, includes available welding quality, control information, system component status; has 164 fields, 40.5 million records,
  - *error protocol*: with minor errors relevant to possible quality deterioration or inefficiency; has 10 fields, 1 million records,
  - *failure protocol*: with more severe quality failures, has 24 fields, 168 thousand records,
  - *change protocol*: with recorded manual interference by operators; has 16 fields, 272 thousand records.
- *feedback curves database*: with sensor data of resistance, pulse width modulation, force, etc., measured per millisecond during the welding process; has 14 fields, 2.7 million records.
- *reference process curves database*: with the target feedback curves prescribed by welding programs; has 14 fields, 196 groups corresponding to 2.7 million records of the feedback curves.
- *meta settings database*: with general configurations of welding sheet material, geometry, adhesive, etc.; has 23 fields, 196 groups corresponding to 40.5 million records.

These raw data has various formats, including SQL database, Excel tables, text files, RUI-files and can have many discrepancies in variable names and data formats. Thus, we merged different protocols, databases, etc., unified data formats, variable names, and transformed them into one uniform data format. This process of data preparation and integration was time-consuming and resulted in 263 fields, 53.2 million records, and 1.4 billion items in total.

Our next step was to understand what data is more important. We organised several workshops with welding process experts and selected and prepared fragments of data that correspond to two representative welding machines, Welding Machine 1 (WM1) and 2 (WM2) that perform two and four welding programs respectively. These *integrated data* correspond to 2.74 million records and 44.61 million items and capture 1998 and 3996 welding operations of WM1 and WM2 respectively.

These data are comprised of features in two levels:

- Data on the *welding time level*, which contain 4 meaningful *Process Curves*, including electric current (I), voltage (U), resistance (R), pulse width modulation (PWM). They are measured per millisecond, and are of different lengths, ranging from 400 to more than 1000 samples, depending on the actual welding time. These process curves form *Time Series* (TS) on the welding time level.
- Data on *welding operation level*, which contain 188 meaningful *Single Features* (SF). They are constants for each single welding spot. The consecutive single features form time series on the welding operation level. More precisely, these single features are:

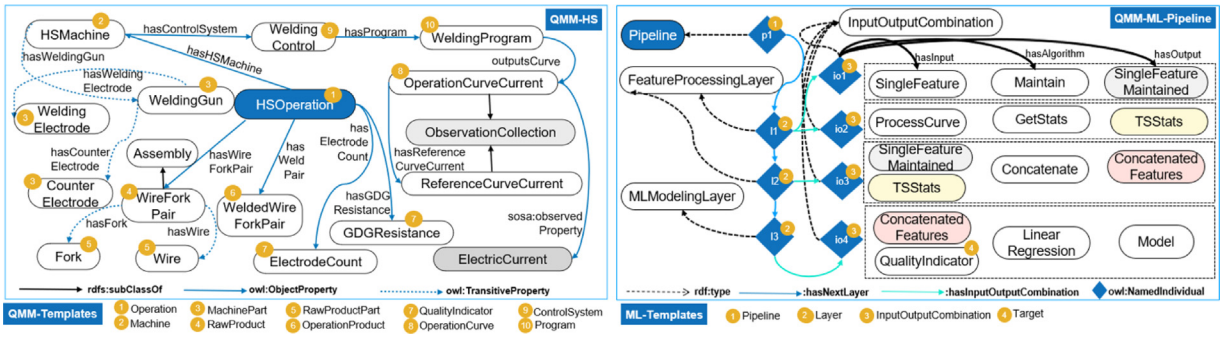

Fig. 9. Schematic illustration of examples of HS ontology and ML pipeline ontology. Some object properties and classes are omitted for simplicity. The rectangles with dashed border indicate that the named individuals of InputOutputCombination are connected to instances of the classes in the rectangles.

- *Program Numbers (ProgNo)* are nominal numbers of the welding programs, each prescribing a set of welding configurations.
- *Count Features*, including *WearCount*, which records the number of welded spots since last *dressing*,[4] *DressCount*, which records the number of dressings performed since last cap change, and *CapCount*, which records the number of cap changes.
- *Status*, describing the operating or control status of the welding operation, e.g. System Component Status, Monitor Status, and Control Status.
- *Process Curve Means*, which are the average values of the process curves during their welding stages, calculated by the welding software system.
- *Quality Indicators*, which are categorical or numerical values describing the quality of the welding operations, e.g. Process Stability Factor, HasSpatter, and the output feature *Q-Value*.

4.3. Problem definition

The quality monitoring task is to maintain the Q-Value as close to 1 as possible for all welding spots during manufacturing. In practice, we would like to do it by learning estimations of Q-Values before the actual welding happens and then to take preventive actions if the predicted value is too low: change parameters of welding machines, replace welding caps, etc. More formally, we need an estimation function f mapping manufacturing data to the Q-Value of the next welding operation Q_{next} as: $Q_{next} = f(X_1, \ldots, X_{prev-1}, X_{prev}, SF^*_{next})$, where $X_1, \ldots, X_{prev-1}, X_{prev}$ include data (single features and time series) of previous welding operations and known features of the next welding operation (SF^*_{next}, e.g. welding program).

5. Demonstration of the semantic software on the use case

We now demonstrate how SemML is applied on the use case, including the domain ontologies, ML pipeline ontology, detailed mechanism of semantic enhanced ML and a short introduction of the ML pipelines in the use case.

5.1. Domain and application ontologies

We now present two domain ontologies and several application ontologies that were used in our use case.

QMM-Resistance Spot Welding Ontology. By applying our templates, Bosch domain specialists created the `QMM-RSW` – ontologies for the resistance spot welding process. `QMM-RSW` created by different users for different datasets differ in small details. A typical example ontology features 1542 axioms, which define 84 classes, 123 object properties and 246 datatype properties and can be expressed using $\mathcal{SH}(\mathcal{D})$ Description Logics. An example of `QMM-RSW` and its relationship to templates are partially shown in the right part of Fig. 5.

QMM-Hot-Staking Ontology. Bosch domain specialists also created ontologies for the hot-staking welding process. Similar to `QMM-RSW`, `QMM-HS` ontologies created by different users differ in details. Still, all of them can be expressed using $\mathcal{SH}(\mathcal{D})$ Description Logics. Templates guarantee that new domain ontologies will not exceed the desired expressivity. An example `QMM-HS` ontology features 1491 axioms, which define 64 classes, 90 object properties and 182 datatype properties. *QMM-HS* and templates are partially shown in the left part of Fig. 9. Hot staking bears resemblance to the RSW process: a *qmm-hs:HSOperation* is performed by a *qmm-hs:HSMachine* on a *qmm-hs:WireForkPair*. It results in a *qmm-hs:WeldedWireForkPair*. For the hot staking process, *qmm-hs:GDGResistance* and *qmm-hs:ElectrodeCount* are established quality indicators. The *qmm-hs:WeldingControl* unit of the *qmm-hs:HSMachine* monitors multiple operation curves. Ontologies for HS showed that users can successfully apply QMM-Templates to create ontologies for similar manufacturing processes. Moreover, for hot staking we could largely reuse the modelling of operation curves from the RSW process.

QMM-ML Pipeline Catalogue. Four ML pipelines were developed beforehand and encoded in ontologies (details see Section 5.3 and Figs. 9 and 11). These ML pipelines differ in their layered architecture, preprocessing and feature engineering strategies, ML algorithms used and the expected input data. QMM-ML Pipeline catalogue stores the configurations corresponding to each ML pipeline as ontologies.

QMM-ML-Pipeline Ontology. The four ML pipelines are encoded in ML pipeline ontologies and can be tailored to specific datasets by SemML. In the right part of Fig. 9, we present an example ML-Pipeline Ontology, which encodes the simplest ML pipeline Base-LR (Section 5.3). We omit the *PreparedDataLayer*, which only contains three feature groups, *FG-SingleFeature*, *FG-ProcessCurve* and *QualityIndicator*. These 3 groups must be present in the dataset for the pipeline to guarantee a good performance. Fig. 9 exemplifies a series of *FeatureProcessingLayer*s and ends with *MLModellingLayer*. Each layer has input, output, and the algorithm to be applied to the input. It also points to the next layer to be executed. Often, the output of the previous layer becomes the input to the next layer.

5.2. Semantically-enhanced machine learning

We now explain how we semantically enhanced ML with ontologies and novel processing modules by following the workflow in Fig. 10, where the Step 2 and Step 3 are illustrated in detail.

[4] *Dressing* is a type of maintenance operation in RSW, that is to remove a very thin layer of the electrode cap surface to restore the surface condition of the cap to a starting condition that is more suitable for welding.

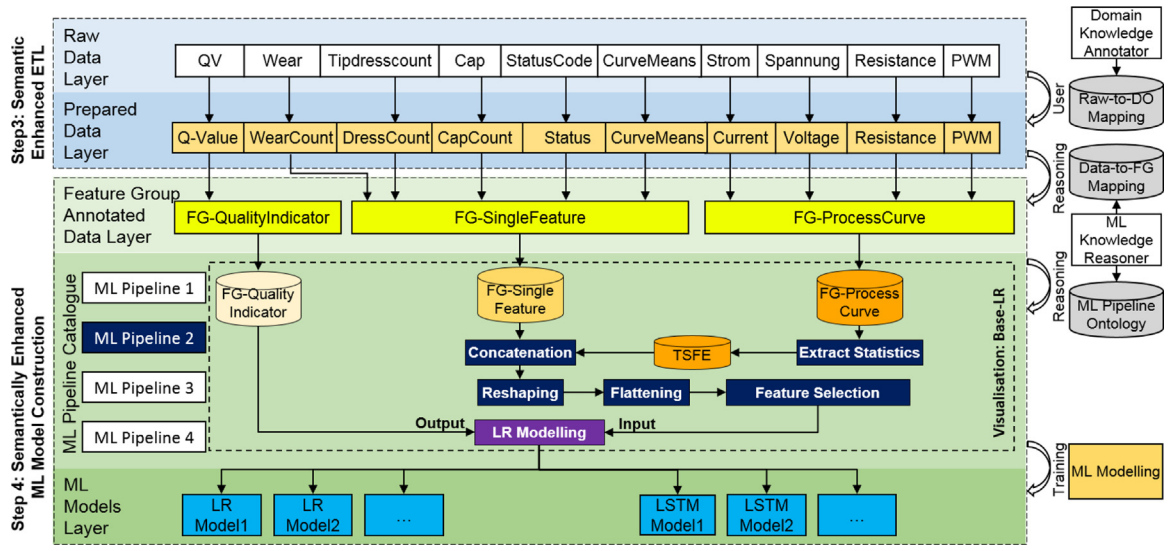

Fig. 10. Mechanism of semantic enhanced ML model construction in "Static Mode": data preparation and ML pipeline selection from a catalogue without dynamically changing the ML pipeline structures.

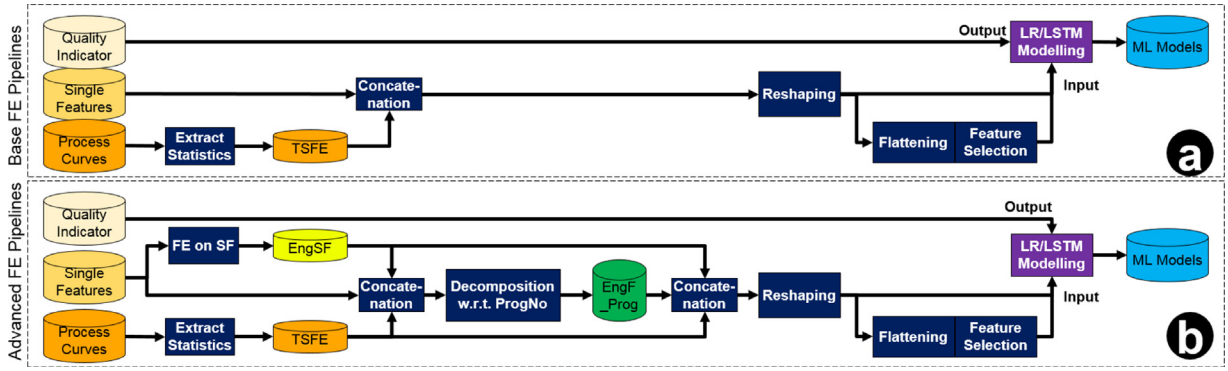

Fig. 11. Visualisation of ML pipelines of *Base* and *Advanced* Feature Engineering (FE). LR: linear regression, LSTM: long short-term memory network, TSFE: time series features engineered, SF: single features, EngSF: engineered single features, EngF_Prog: engineered features with respect to program numbers.

In the raw data layer, several example features in the raw data in ML database are shown, e.g. QV, Wear, Tipdresscount (Fig. 10). The raw data are first annotated by users with domain terms, generating the Raw-to-DO (raw to domain) mapping. Using this mapping, the data are processed by the ETL module and turned into the prepared data (prepared data layer). The feature names are changed to unified feature names, e.g. Q-Value, WearCount, DressCount. Since all features are connected to domain ontologies through the domain term annotation, they are also connected to *QMM-Core*, because domain ontologies are constructed based on *QMM-Core*. Then, these features are also connected to the ML ontology. Through the ML Knowledge Reasoner, the *Feature Groups* (FG) can be automatically reasoned and used as annotations for the features (Feature Group Annotated Data Layer in Fig. 10).

We now illustrate this reasoning with an example:

(18) Class: *qmm-rsw:ElectrodeCap*

(19) SubClassOf: *qmm-core:SystemComponent*

(20) SubClassOf: *qmm-rsw:hasElectrodeCapStatus* only *qmm-rsw:WearCount*

(21) Class: *qmm-rsw:WearCount*

(22) SubClassOf: *qmm-core:ToolWearingStatus*

(23) Class: *qmm-core:ToolWearingStatus*

(24) SubClassOf: *qmm-core:Status*

(25) SubClassOf: *qmm-core:hasMLFeatureGroup* only *qmm-ml:FG-Wear*

Axiom 20 defines that (the elements of the class) *qmm-rsw:ElectrodeCap* can only have the status parameter *qmm-rsw:WearCount*. The latter has a superclass *qmm-core:ToolWearingStatus* from the Core ontology (Axiom 22), which is assigned the only associated machine learning feature group *qmm-ml:FG-Wear* (axiom 25). Observe that from Axioms (22) and (25) one can derive using reasoning that *qmm-rsw:WearCount*'s only feature group is also *qmm-ml:FG-Wear*, formally:

(9) Class: *qmm-rsw:WearCount*

(10) SubClassOf: *qmm-core:hasMLFeatureGroup* only *qmm-ml:FG-Wear*

The users then view the available ML pipelines and select one pipeline that he thinks suitable to solve the ML task. In the ML pipeline catalogue in Fig. 10, on the left hand side four ML pipelines are illustrated for users to select. The user selects ML Pipeline 2. On the right hand side, ML Pipeline 2 (Base-LR, see Section 5.3) is schematically illustrated, where *FG-QualityIndicator* is used as output of linear regression (LR) modelling, and the *FG-SingleFeature* and *FG-ProcessCurve* are processed by a series

of steps and in the end used as input for the ML modelling. Depending on the ML pipelines, a series of LR models or LSTM models can be generated.

To activate the ML pipeline selected by the user, the feature group annotations are used by the ML knowledge reasoner to automatically map features in the ML database to *Feature Groups* in the ML pipeline ontology that corresponds to the selected ML pipeline. The selected ML pipeline ontology is then preprocessed by the ML knowledge reasoner to be adjusted to the feature groups available in the ML database. For example, if there exist only single features in the ML database, but the selected ML pipeline also contains feature processing steps for *FG-ProcessCurve*, then these steps relevant to *FG-ProcessCurve* will be trimmed. The adjusted ML pipeline ontology then is used to activate the parametrised ML pipeline. The ML pipeline is a set of ML scripts with interface open to feature groups (input to the ML pipeline) and ML models (output of the ML pipeline). By feeding the features in the ML database to the ML pipeline and activating the interface of feature groups, ML models are trained and output to the ML database (ML Models Layer in Fig. 10).

5.3. Four ML pipelines for the use case

In the use case, four ML pipelines were developed beforehand and stored in a catalogue for users' selection. They have been designed in a general manner and evaluated with a large dataset collected from resistance spot welding (RSW) plants [31]. In the following, we first briefly introduce the four ML pipelines, and then present subjective evaluation of semantic enhanced machine learning.

The ML pipelines adopt the classic ML school of feature engineering and modelling [41]. Feature engineering is to manually design some strategies to extract new features from the raw features (named as engineered features) [42]. ML models are then trained on the engineered features. The combinations of two feature engineering strategies, *Base* feature engineering and *Advanced* feature engineering, and two ML models, linear regression and LSTM result in four ML pipelines. Therefore, the four ML pipelines can be denoted as Base-LR, Base-LSTM, Advanced-LR, and Advanced-LSTM.

Base Feature Engineering. As illustrated in Fig. 11.a, Time series features (TS) of different lengths are first padded with different values that are physically meaningful. Current, voltage and pulse width modulation are padded with zero, since after welding they are de facto zero, while resistance is padded with the last value, for resistance is the intrinsic property of matter and does not disappear after welding. After that, eight statistic features, including minimum, maximum, minimum position, maximum position, mean, median, standard deviation, and length, are extracted from the time series (named as time series features engineered, TSFE). These features are then concatenated with single features. The concatenated features are then reshaped to take the certain look-back length of previous welding data for forecasting future welding quality.

Advanced Feature Engineering. As illustrated in Fig. 11.b, three new features are engineered based on single features (named as engineered single features, EngSF): 1) *WearDiff* is calculated as the difference between WearCount of two consecutive welding operations, characterising the degree of change of wearing effect; (2) *NewDress* will be ONE after each dressing, and ZERO otherwise; (3)*NewCap* will be ONE after each Cap Change, and ZERO otherwise. The EngSF are concatenated with RawSF and TSFE, and further processed by *Decomposition with respect to ProgNo*, resulting in engineered features with respect to program numbers (EngF_Prog). The *EngF_Prog* incorporate information of program numbers by decomposing the concatenated RawSF, EngSF and TSFE, which form time series on the welding operation level, to sub-time-series with respect to ProgNo (Fig. 12.a). Each sub-time-series only belongs to one ProgNo. The EngF_Prog include *RawSF_Prog*, *EngSF_Prog*, and *TSFE_Prog*. After that, the RawSF, EngSF, EngF_Prog and TSFE are then again concatenated and reshaped.

Table 1
Model performance tested on test sets.

Data preprocessing	Modelling	mape (WM1)	mape (WM2)
Benchmark: $\hat{Q}_{next} = Q_{prev}$		3.19%	7.74%
Base feature engineering	LR	2.38%	2.50%
	LSTM	2.35%	2.27%
Advanced feature engineering	LR	1.61%	2.10%
	LSTM	2.04%	1.94%

ML Modelling. Following the feature engineering, two ML methods are used for ML modelling. (1) The reshaped features needed to be flattened, and reduced by feature selection. After that, they can modelled by linear regression (LR). LR is selected to demonstrate that even simple ML algorithm can have great performance with meaningfully engineered features. Least squares method is usually used for solving LR modelling. (2) The reshaped features can be directly modelled by LSTM networks. LSTM is an artificial recurrent neural network (RNN) architecture. It is especially suitable for modelling data with temporal dependencies [49].

Hyper-parameter Tuning. The datasets were split into training, validation and test set. The four types of ML models on the training set and the hyper-parameters were selected based on the performance on the validation set. The *number of selected features* and *look-back length* were first selected with Advanced-LR with the performance metric mean absolute percentage error, *mape*, computed as $\frac{1}{N}\sum_{i=1}^{N}|Q_i - \hat{Q}_i|/Q_i \times 100\%$. They are then fixed for other pipelines. The *number of selected features* and *look-back length* will influence the amount of data delivered to ML models. They are therefore first selected with Advanced-LR and then fixed for other pipelines, for making a fair comparison, emphasising influence of feature quality, rather than of data volume caused by these two features. Besides, to make the ML pipelines general for more datasets, these hyper-parameters should provide relatively good performance (not necessarily the best performance), avoid overfitting, and ideally model performance should be insensitive to the hyper-parameters. Several hyper-parameters of the LSTM neural networks are selected after a series of experiments and then fixed because they generally show better performance than their alternatives: *Adeldelta* optimiser, *with* data shuffling, *stateless* LSTM, and *saving the best model* strategy for determining iteration. The number of layers and number of neurons in each layer were selected with limited grid search 12. It can be clearly seen from Fig. 12.b that after around 12 of selected features and a look-back length of 5, the performance of models reaches a plateau (or basin in sense of mape). We have chosen a look-back length of 10 and 20 selected features, for in this area the models are rather insensitive to hyper-parameters.

Performance of Prediction Accuracy. The results for the four models are summarised in Table 1. The performance of the ML pipelines is better than the benchmark, indicating effectiveness of feature engineering. The model performance improves from *Base* to *Advanced* as expected: the higher level of feature engineering brings improvements. The best model for WM1 is Advanced-LR (1.61%), while for WM2 Advanced-LSTM (1.94%). Advanced-LR outperforms Advanced-LSTM for WM1, while for WM2 it is the other way around. We postulate that the reason is that WM1 has less complex data than WM2.

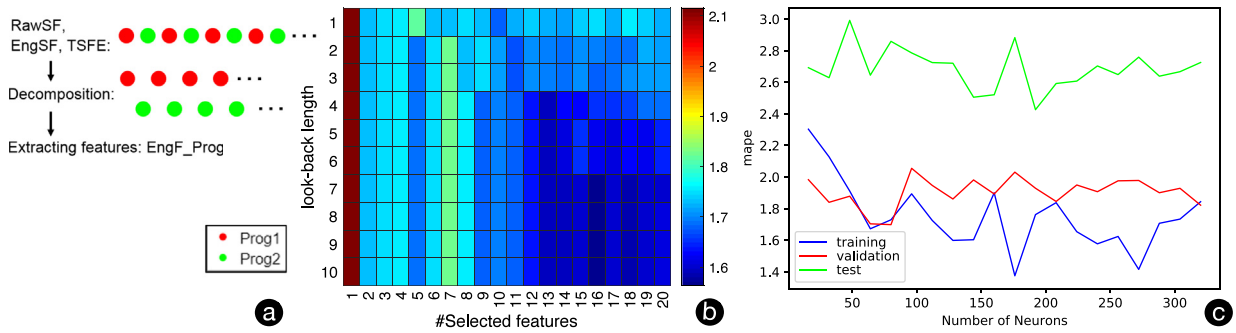

Fig. 12. (a) Generating EngF_Prog by decomposing RawSF, EngSF and TSFE to sub-time-series on the welding operation level; (b) Selecting the *look-back length* and *#Selected features* using mape on validation set; (c) Selecting the *#Neurons* of LSTM [31].

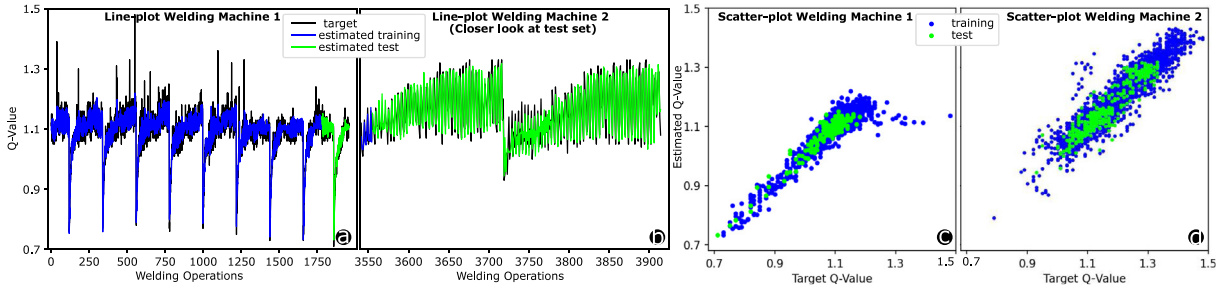

Fig. 13. Prediction results of Advanced-LR for WM1 in (a) line and (c) scatter plots and for WM2 in (b) line and (d) scatter plots [31].

The best prediction results for WM1 are by Advanced-LR and for WM2 are by Advanced-LSTM. Their prediction results are illustrated in Fig. 13 with line and scatter plots.

6. Evaluation with user study

For evaluation of the Ontology Extender and Domain Knowledge Annotator, we conducted a user study with 14 Bosch experts. These two components essentially addresses the challenges C1 on communication and C2 on data integration by. Since the SemML runs in a "Static Mode" for ML construction, the users only need to select ML pipelines without dynamically changing the ML pipelines, and this will not affect the ML prediction performance. Therefore, we could only evaluate the ML Knowledge Reasoner and ML Interpreter in a subjective manner. More extensive evaluation remains our future work. These two components address C1 on communication and C3 on ML generalisability. In the subjective evaluation, users gave positive feedbacks on reduced time for process and data understanding. The meetings required for understanding new datasets were reduced from approximately 18 meetings to 8 meetings. They comment that when new datasets arrive, they do not need to dive into the data level and repeat the complete ML construction process again. Instead, they only need to annotate new datasets with domain ontologies, or extend the domain ontologies if needed.

To the end of evaluation of Ontology Extender and Domain Knowledge Annotator, we organised a workshop with three parts. First, we organised a thirty-minutes crash course to explain the ontology `QMM-Core` and templates. Then, we conducted two experiments: Experiment 1 on *Ontology Extension*, where the users were asked to describe their domains in terms of `QMM-Core` by filling in the proposed templates, and Experiment 2 on *Data Mapping*, where the users were asked to map the variables in the raw data sources to the datatype properties in the ontologies they created. Note that our experiments do not aim at comprehensive coverage of the welding domains and data sources relevant for welding quality: in our evaluation tasks we tried to balance the coverage and the time required to accomplish them.

6.1. Design of experiments

We give further details on experiments and participants.

Users. Two target user groups (with the roles of domain experts and data scientists) participated in the experiments with two welding processes: resistance spot welding (RSW) and hot-staking (HS). The users could choose to participate only in Experiment 1 or in both. Some of them took part in the experiments with more than one domain or role. This is the case, e.g., for users who are domain experts both for RSW and HS, and some users who are domain experts but are learning data analysis or vice versa. In total, from 14 participants 25 result instances were collected in Experiment 1, and 19 instances in Experiment 2. Before the experiments, the participants rated their domain expertise (E1), experience with semantic technologies (E2) and experience with data mapping tools (E3) on a Likert scale (1: Beginner, 2: Developing, 3: Competent, 4: Advanced, 5: Expert).

Experiment 1: Ontology Extension. The users were asked to use Ontology Extender to create their ontologies. As illustrated in Fig. 14.1.1, for each term highlighted with the blue background in the short descriptions for the welding processes on the left side, the users selected a template on the right side, and then made choices to link the created class to its dependencies (drop-down list in Fig. 14.1.2). The resulting ontology terms (classes and properties) were then visualised (Fig. 14.1.3). Note that domain experts and data scientists did their tasks sequentially: the former created an ontology, and then the latter inspected their ontologies and extended them with quality indicators.

Experiment 2: Data Mapping. As illustrated in Fig. 14.2, the users were asked to use Domain Knowledge Annotator to map data. For each term in the column of raw variable names on the left side, they clicked the group of classes from the right top panel, selected a class, and then chose the datatype properties where the class is a domain from a drop-down list (in the right bottom panel).

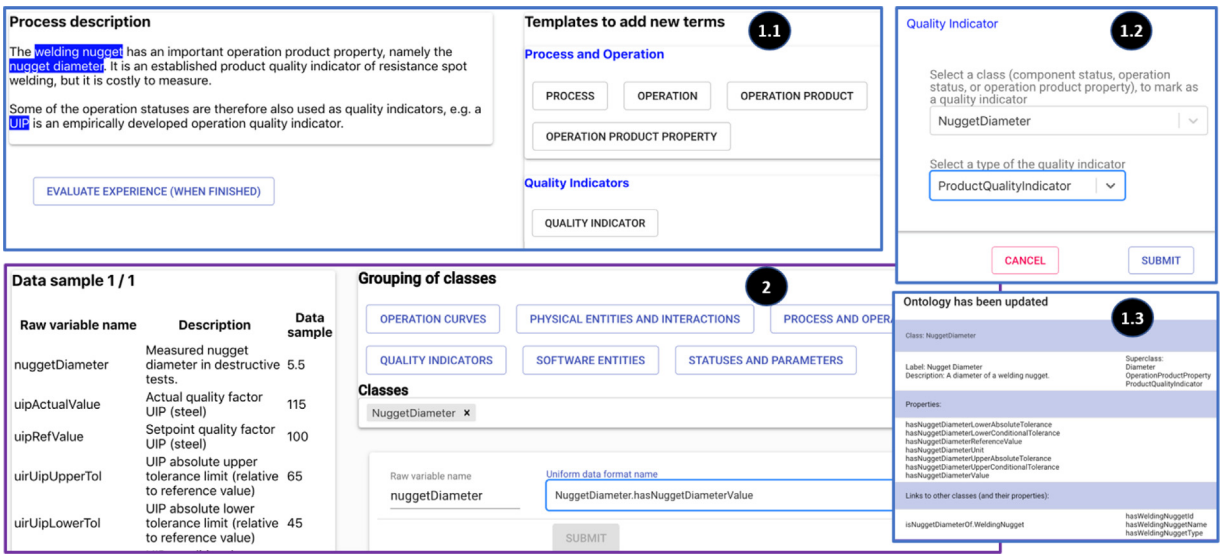

Fig. 14. Graphical user interfaces for (1.1–1.3) Ontology Extension and (2) Data Mapping [10].

6.2. Evaluation metrics

According to ISO 9241-11 system usability has 3 dimensions: *effectiveness*, *efficiency*, and *satisfaction* [50]. We rely on them and their correlations with user expertise.

Effectiveness shows to which extent the intended goal is achieved [50]. We use *correctness*, the percentage of successfully completed tasks, as the metric for it. We are fully aware that there is no absolute correctness for these tasks because the domains or data can be understood in different ways. This issue is however not critical in our experiments since we carefully designed the tasks so that the answers are minimally controversial across the experts. In Experiment 1, the correctness is defined as the percentage of correctly chosen templates for a given term (Template Correctness, *TC*), the percentage of correct choices linking the dependencies between classes (Choice Correctness, *ChC*), and the percentage of fully correctly created classes, for which the correct template is chosen and all dependencies are correctly specified (Final Correctness, *FC*). In Experiment 2, the correctness is defined as the percentage of correctly chosen classes (Class Correctness, *ClC*) and the percentage of correctly mapped datatype properties for each item of raw variable names (Item Correctness, *IC*).

Efficiency corresponds to "the resources (such as time or effort) needed by users to achieve their goals" [50]. We use *time spent on tasks* as the metric of efficiency.

Satisfaction was evaluated with the questionnaires after each experiment on 6 dimensions (see Table 2): [D1] *User Friendliness*: the system is easy to use; [D2] *Self-Explainability*: the system does not require extra knowledge or support; [D3] *Consistency*: the system is consistent in format, workflow, wording, etc.; [D4] *Completeness*: the system covers the domain/data to describe/understand; [D5] *Descriptive Power*: the system allows to describe the domain/data effectively, clearly; [D6] *Communication Easiness*: the system eases the communication between experts. The questions are presented in Table 2 where the first five are inspired by System Usability Scale (SUS). Questions 1–5 (and the corresponding Dimensions 1–3) target the usability of the graphical user interface. They are identical for all roles and experiments. Questions 6–10 (Dimensions 4–6) address more specific issues and differ slightly with respect to the role or the experiment. The users were asked to give scores ranging from 1 to 5 with a Likert scale (1: Strongly disagree, 2: Disagree, 3: Neither agree or disagree, 4: Agree, 5: Strongly agree). Note that Questions 2–5 are negatively formulated. Their scores are reversed in the later analysis to make the representation of the results more intuitive and consistent. E.g. if a user scores Q2 with 1, which means the user strongly disagrees that the system is complex, the corresponding score is reversed to 5, indicating the system is not complex.

6.3. Evaluation results and discussion

The results of the Effectiveness and Efficiency metrics are summarised in Fig. 15, which shows the user performance on Ontology Extender (Fig. 15.1– 15.3) and Domain Knowledge Annotator (Fig. 15.4).

Results for Experiment 1: Ontology Extension. Domain experts in RSW created 14 terms and those in HS created 15 terms. Data scientists created 2 terms for both processes. On average, the users needed about 50 s to create a new term. Note that the description of one term adds from 4 to 25 classes and properties to the ontology (see the process exemplified in Fig. 6 of Section 3.4).

Some users needed extra time for the terms WeldingMachine, CapWearCount and CapDressCount (with high standard deviation shown in the figures). The potential reasons are that the users needed to understand the complex structure of machine and its multiple parts; for the latter two terms (both are created by the SystemComponentStatus template in Fig. 15.3) the users specified two dependencies, which is one more than the normal case of one choice. Another reason could be that the users moved to a new template group, which increased the cognitive complexity of the task and thus, the time spent on the task. In line with this tendency, we observe a gradual decline in time for subsequent terms created with the same or similar templates. For example, WeldingRobot and WeldingGun are machine parts directly following the WeldingMachine, and CapDressCount directly follows CapWearCount. This strongly supports the learnability of the system: having experience with a template increases efficiency and effectiveness.

The average correctness for applying a template is 93%, for making choices of the dependencies is 92%, and for both (final correctness) is 90%. The terms, e.g. CapWearCount, that required more time to create often have a relatively low correctness ratio. The high average correctness strongly demonstrates the usability and the error prevention potential of the system.

Fig. 15. 1 and 2: User performance of time and correctness for RSW in 1 and for HS in 2, aggregated on template groups for both in 3. Time and correctness for data mapping in 4 [10].

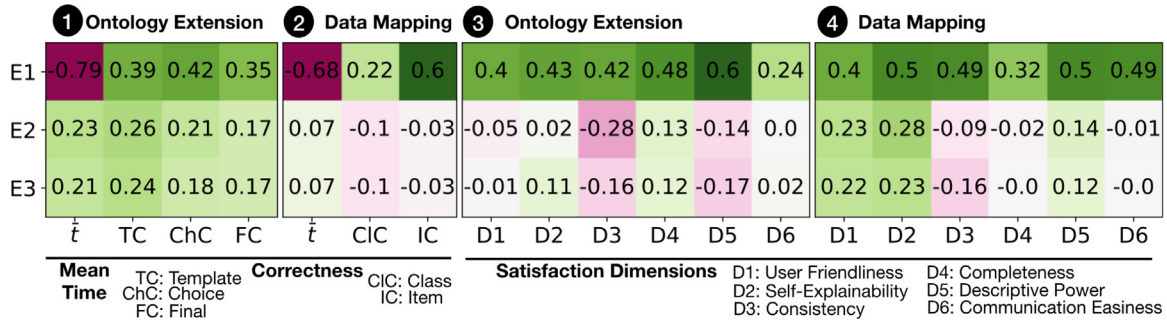

Fig. 16. Heatmaps of correlation coefficients between the usability metrics and self-accessed expertise. E1: Domain expertise, E2: Experience with semantic technologies, E3: Experience with data mapping tools [10].

One of the goals of Ontology Extender was to serve as the communication platform between domain experts and data scientists. In our experimental setup, the data scientists were supposed to (1) inspect the domain ontologies created by the domain experts and (2) add the terms relevant for quality analysis. In particular, they had to add or find a term, and characterise it as a quality indicator. We separate these two parts by the vertical dashed lines

Table 2
Satisfaction metrics: Questionnaires and aggregated quality dimensions.

Experiment 1: Ontology extension		Experiment 2: Data mapping		Dimension
Domain experts	Data scientists	Domain experts	Data scientists	
Q1: I felt very confident using the system Q2: I found the system unnecessarily complex				D1: User friendliness
Q3: I needed to learn many things before I became productive with this system Q4: I needed support of a technical person to be able to use this system				D2: Self-explainability
Q5: I thought there was too much inconsistency in this system				D3: Consistency
Q6: I think the system covers most of my fundamental requirements for the *				D4: Completeness
* Description of the process	* Understanding of the process	* Description of the data	* Understanding of the data	
Q7: I think the system/resulting model allows me to unambiguously *				D5: Descriptive power
* Describe the process	* Understand the process	* Describe the data	* Understand the data	
Q8: I think the relevant aspects of process quality are well presented in the *		I think the mapping system saves me effort of *		
* System	* System and the resulting model	* Describing the data	* Understanding and completing the data mapping	
Q9: I think the resulting ontology/mapping would be very easy to understand for *				D6: Communication easiness
* Data scientists	* Domain experts	* Data scientists	* Domain experts	
Q10: I think the resulting ontology/mapping provides a good common base for discussion with *				
* Data scientists	* Domain experts	* Data scientists	* Domain experts	

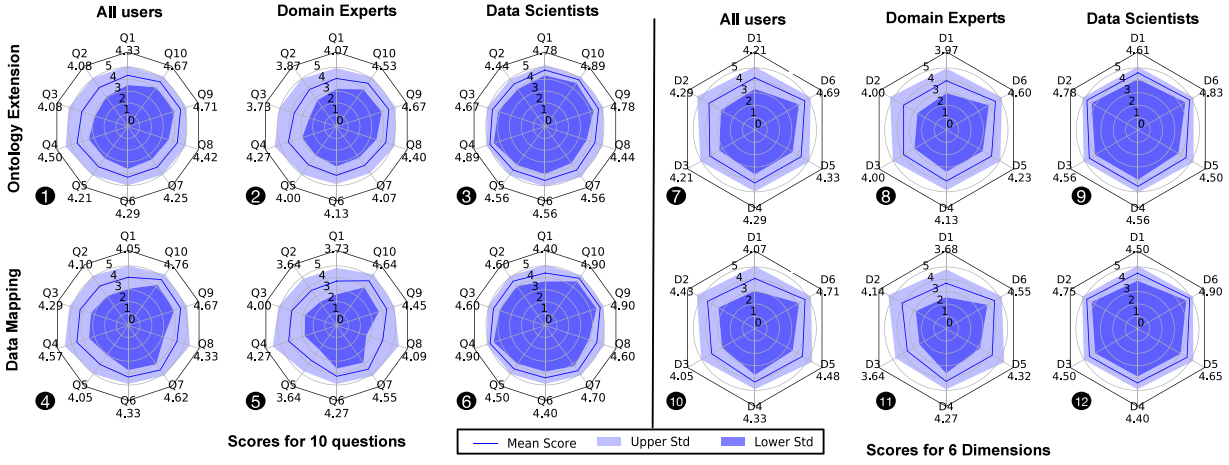

Fig. 17. Radar charts of questionnaires scores on 10 questions (1–6) and aggregated to 6 dimensions (7–13) defined in Table 2. Std is standard deviation. The blue lines indicate the mean scores, the light blue shadow the mean + std, and the dark blue shadow the mean - std. [10]. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

in Fig. 15.1– 15.3. All data scientists achieved 100% correctness with an average time of 39 s.

We now analyse the correlations between the self-reported expertise of our users and their performance. Fig. 16.1 shows a strong negative correlation between domain expertise (E1) and time t and relatively strong positive correlation between E1 and the three types of correctness (template, choice and final). Not surprisingly, the users with higher domain expertise provided more correct modelling solutions and were faster than the beginners. The figure also suggests insignificant correlation between the performance of users and their experience in semantic technologies and mapping tools. This is encouraging since it suggests that the usage of our system requires no or little prior training in these disciplines and activities.

Results for Experiment 2: Data Mapping. The majority of users correctly mapped column names in the suggested files to the newly introduced terms, achieving 100% correctness (Fig. 15.4). The average time they spent for each term is about 50 s. The correlations in Fig. 16.2 support the idea that domain expertise will ease the work and the other two parameters, including the self-accessed experience with mapping tools, have almost no effect. We interpret it as the evidence that the system is able to serve as a solution for both tasks – data modelling and data mapping – and does not require any prior experience with similar technologies. We complement our analysis with the results of the satisfaction questionnaires in the following section.

Satisfaction. We report the satisfaction results in the radar charts in Fig. 17 separately for data scientists and domain experts, and aggregated for all users. The charts in Figs. 17.1 and 17.4 represent the average scores for both user groups. These scores are higher than 4, which indicates a general good impression of users. The mean scores on Questions 4, 9 and 10 are very high ($>$ 4.5): The users evaluate the tool as easy to use without support of a technical person, and they think their working results will be easy to understand for other experts. This supports our vision that an ontology can serve as a good communication base.

The comparison of the scores on questions by the domain experts and data scientists (Figs. 17.2 vs. 17.3, 17.5 vs. 17.6), reveals that the data scientists evaluate the system with higher scores in average, and smaller standard deviation. This indicates

that the data scientists have better and more uniform opinions on the system, while users taking the role of domain experts have more diverse opinions. One reason for that could be that the tasks for data scientists were only related to quality indicators and thus more clearly defined, while the tasks for domain experts who needed to describe complicated processes were more demanding.

In Figs. 17.7–17.12 the scores on questions are aggregated to six dimensions (see the meanings of dimensions in Table 2). This aggregation makes it easier to draw conclusions. Firstly, all six dimensions have average scores over 4, which means the users are satisfied with the system in general. The scores for D6 (communication easiness) and D5 (descriptive power) are the highest, indicating the users appreciate the ease of communication. Dimensions more related to the system usability (D1–D3) have scores around four, which means there is improvement space for the user interface.

Correlations in Figs. 16.3 and 16.4 reveal similar results as for the performance analysis: domain expertise correlates with high satisfaction scores, while the other two areas of expertise have little effect, which supports that the tool requires little prior training.

7. Related work

Survey [51] extensively covers the usage of semantic technologies in data mining and knowledge discovery, and in particular in the facilitation of machine learning workflows. Still, to the best of our knowledge, existent approaches and system solutions, including the recent developments of digital twins for manufacturing [52], only partially meet our requirements R1–R5. Thus we had to develop our own ontologies and templates as well as ontology-based, highly customised and configurable solution, integrated into the workflow to support quality analysis in manufacturing. The users of our system are the different experts responsible for the task of developing machine learning methods. Indeed, none of the ontologies for manufacturing (e.g., [53–58]) fully serve as the communication model for our use cases and sufficiently cover our domains. The mapping-based data integration solutions like Ontop [59] are not particularly targeted towards our aim of minimising the involvement of ontologists into the model maintenance processes. Moreover, the role of mappings in our context is not limited to the transformation of data into the RDF format. Firstly, we integrate data sources for machine learning, secondly and in line with these tools, we transform some parts of it to RDF to explore the data. In the metadata management solutions for data lakes like Constance [60] and GEMMS [61], the metadata descriptions are used to integrate the raw sources. As the mapping-based data integration solutions, these systems lack the extensibility aspect. We found that the existing tools for ontology extension, e.g. template-driven systems (Webulous [62], TermGenie [63], Ontorat [64]) required considerable adjustments (including but not limited to the development of the new graphical user interface) and could not be easily integrated with the machine learning workflow and our infrastructure. Thus, we developed our own ontology extension tooling.

Machine learning for quality monitoring in RSW has been focusing on estimating the spot diameters after welding [65,66]. Most previous approaches analysed data collected from limited laboratory experiments [67,68]. Ontology in welding has studied the incoherence of standards [69], or was used for ML modelling with semantics [66]. Other works relating ontology to data analysis have discussed improving data understanding [70], or database access with annotations [14,15]. A closer work [71] discussed domain integration and explainability. Many studies tried to combine ML and semantic technologies. The work [72] attempted to combine ontologies and ML by relying on the topic categories that regulate the emission of disclosures to improve the ML classifiers. Another work [73] combines ML and semantic orientation with a voting system based on the majority rule for classifying opinions to positive or negative. A review [74] summarises common approaches of combining semantic web technologies and ML, such as mapping network inputs or neurons in supervised learning to classes in ontologies or entities in knowledge graphs [75,76], creating explainable knowledge embeddings to increase explainability for unsupervised learning [77,78], relying on an ontology with extra information to create rules which limit the number of recommended actions in reinforcement learning [79]. None of these works addressed generalisability and extensibility to new domains and the application of ontologies for feature engineering.

8. Lessons learned, conclusion, and outlook

Lessons Learned. First, an ontology as a formal language can be very effective to provide a lingua franca for communication between experts with different knowledge backgrounds. The process of developing the Core model was onerous, time-consuming and cognitively demanding at the initial phase. After that, we had a basis of core model and a set of templates. It revealed the development process became much easier because the developed ontologies facilitated communication. Second, the technology of templates enables non-ontologists to describe their domains and data in a machine-readable and unambiguous way. In contrast to the simple, tabular interfaces which exist for the template-based ontology construction, we needed to address the new requirements for our use case. In the use case, the users need to generalise a series of classes, while the later generated classes have dependencies on the older ones. This has two requirements: (1) the newly generated classes need to be accessible to later generated classes; (2) sequences of templates need to be applied in a particular order because the later classes presuppose the existence of their depending classes. Furthermore, the users need assistance like drop-down lists and visualisation of changes. Third, the users that are unfamiliar with the domains will need more time for some tasks and yield lower correctness. This indicates that we need to split the iterative process of task negotiation to smaller units so that different experts can digest each other's information more smoothly.

Conclusion. In this work we presented a ontology-based software architecture SemML that enhances ML analysis for condition monitoring with semantic technologies. SemML addresses three challenges and five requirements, and enhances four steps of the workflow of ML analysis for condition monitoring. SemML allows users with minimal knowledge in semantics and machine learning to construct ML models by presenting the users a set of ML pipelines developed beforehand and automatically adjust the user-selected ML pipeline with smart semantic reasoning. The users only need to do data annotation with the GUI provided by our software. SemML also allows users to extend the ontologies for their domains with good quality, if the datasets require so. SemML consists of four semantic components: Ontology Extender, Domain Knowledge Annotator, Machine Learning Reasoner, and Ontology Interpreter. We implemented the software architecture as the SemML system. We then evaluated SemML on a Bosch use case of welding quality monitoring, focusing on the first two semantic modules. To this end, we conducted a user study with 14 Bosch experts. The evaluations show promising results: SemML can indeed help in addressing the challenges of communication and data integration.

Outlook. We plan to evaluate SemML's third and fourth semantic module with a user study. We also plan to extend SemML to

a "Dynamic Mode", where users will be able to dynamically configure the ML pipelines stored in the ML pipeline catalogue with a GUI even when the users only have minimal knowledge of machine learning. It is in development and under extensive evaluation. An automatic hyper-parameter tuning function module will be added to SemML to further improve the generalisability of ML pipelines. SemML has been deployed in a Bosch evaluation environment, and we plan to further evaluate and strengthen it to eventually push it into the production; moreover, we envision to generalise it over wider range of industries, e.g. oil industry. This in particular requires to show the benefits of SemML with more users and in other use cases. This also requires to further improve the usability of SemML with more advanced services such as access control as well as with various ontology visualisation modules. In particular, we envision to improve the usability by exploring keyword-[80–82] and faceted-search [83–86] over catalogues of ML pipe-lines, as well as with summarisation techniques for ontologies, Knowledge Graphs, and semantically represented ML-pipelines [87–89] and possibly auto-generated text enhancements for ML-pipelines [90,91]. We also consider a possibility to offer dataset search accompanying SemML to help users to select the most appropriate datasets for concrete ML tasks [92–95]. Moreover, we consider developing further visual paradigms and interfaces to improve users experience when interacting with SemML [17,18,96,97]. Finally, we are planning to enhance SemML with semi-automatic support for data annotation [98,99].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was partially supported by SIRIUS Centre, Norwegian Research Council project number 237898. We are thankful to Tim Pychynski for insightful discussions.

References

[1] H. Kagermann, Change through digitization – value creation in the age of industry 4.0, in: Management of Permanent Change, 2015.
[2] ITU, Recommendation ITU-T Y.2060: Overview of the Internet of things, Tech. rep., International Telecommunication Union, 2012.
[3] S. Chand, J. Davis, What is smart manufacturing, Time Mag. Wrap. 7 (2010) 28–33.
[4] W. Thorsten, W. Daniel, I. Christopher, T. Klaus-Dieter, Machine learning in manufacturing: Advantages, challenges, and applications, Prod. Manuf. Res. 4 (1) (2016) 23–45.
[5] B. Zhou, M. Chioua, J.-C. Schlake, Practical methods for detecting and removing transient changes in Univariate oscillatory time series, IFAC-PapersOnLine 50 (1) (2017) 7987–7992.
[6] D. Mikhaylov, B. Zhou, T. Kiedrowski, R. Mikut, A.F. Lasagni, Machine learning aided phase retrieval algorithm for beam splitting with an LCoS-SLM, in: Laser Resonators, Microresonators, and Beam Control XXI, Vol. 10904, 2019, p. 109041M.
[7] D. Mikhaylov, B. Zhou, T. Kiedrowski, R. Mikut, A.-F. Lasagni, High accuracy beam splitting using SLM combined with ML algorithms, Opt. Lasers Eng. 121 (2019) 227–235.
[8] B. Zhou, M. Chioua, M. Bauer, J.-C. Schlake, N.F. Thornhill, Improving root cause analysis by detecting and removing transient changes in oscillatory time series with application to a 1,3-Butadiene process, Ind. Eng. Chem. Res. 58 (2019) 11234–11250.
[9] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, R.X. Gao, Deep learning and its applications to machine health monitoring, Mech. Syst. Signal Process. 115 (2019) 213–237.
[10] Y. Svetashova, B. Zhou, T. Pychynski, S. Schmidt, Y. Sure-Vetter, R. Mikut, E. Kharlamov, Ontology-enhanced machine learning: a Bosch use case of welding quality monitoring, in: ISWC, 2020, pp. 531–550.
[11] E. Kharlamov, S. Brandt, E. Jiménez-Ruiz, Y. Kotidis, S. Lamparter, T. Mailis, C. Neuenstadt, O.L. Özçep, C. Pinkel, C. Svingos, D. Zheleznyakov, I. Horrocks, Y.E. Ioannidis, R. Möller, Ontology-based integration of streaming and static relational data with optique, in: SIGMOD, 2016, pp. 2109–2112.
[12] E. Kharlamov, B.C. Grau, E. Jiménez-Ruiz, S. Lamparter, G. Mehdi, M. Ringsquandl, Y. Nenov, S. Grimm, M. Roshchin, I. Horrocks, Capturing industrial information models with ontologies and constraints, in: ISWC, 2016.
[13] E. Kharlamov, Y. Kotidis, T. Mailis, C. Neuenstadt, C. Nikolaou, O.L. Özçep, C. Svingos, D. Zheleznyakov, Y.E. Ioannidis, S. Lamparter, R. Möller, A. Waaler, An ontology-mediated analytics-aware approach to support monitoring and diagnostics of static and streaming data, J. Web Semant. 56 (2019) 30–55.
[14] E. Kharlamov, D. Hovland, M.G. Skjæveland, D. Bilidas, E. Jiménez-Ruiz, G. Xiao, A. Soylu, D. Lanti, M. Rezk, D. Zheleznyakov, M. Giese, H. Lie, Y.E. Ioannidis, Y. Kotidis, M. Koubarakis, A. Waaler, Ontology based data access in statoil, J. Web Semant. 44 (2017) 3–36.
[15] E. Kharlamov, T. Mailis, G. Mehdi, C. Neuenstadt, Ö.L. Özçep, M. Roshchin, N. Solomakhina, A. Soylu, C. Svingos, S. Brandt, M. Giese, Y.E. Ioannidis, S. Lamparter, R. Möller, Y. Kotidis, A. Waaler, Semantic access to streaming and static data at Siemens, J. Web Semant. 44 (2017) 54–74.
[16] I. Horrocks, M. Giese, E. Kharlamov, A. Waaler, Using semantic technology to tame the data variety challenge, IEEE Internet Comput. 20 (6) (2016) 62–66.
[17] A. Soylu, M. Giese, R. Schlatte, E. Jiménez-Ruiz, E. Kharlamov, Ö.L. Özçep, C. Neuenstadt, S. Brandt, Querying industrial stream-temporal data: An ontology-based visual approach, J. Ambient Intell. Smart Environ. 9 (1) (2017) 77–95.
[18] A. Soylu, E. Kharlamov, D. Zheleznyakov, E. Jiménez-Ruiz, M. Giese, M.G. Skjæveland, D. Hovland, R. Schlatte, S. Brandt, H. Lie, I. Horrocks, OptiqueVQS: A visual query system over ontologies for industry, Semant. Web 9 (5) (2018) 627–660.
[19] Y. Sun, L. Zhang, G. Cheng, Y. Qu, SPARQA: Skeleton-based semantic parsing for complex questions over knowledge bases, in: AAAI-IAAI-EAAI 2020, 2020, pp. 8952–8959.
[20] M. Ringsquandl, E. Kharlamov, D. Stepanova, M. Hildebrandt, S. Lamparter, R. Lepratti, I. Horrocks, P. Kröger, Event-enhanced learning for KG completion, in: ESWC, 2018.
[21] E. Kharlamov, G. Mehdi, O. Savkovic, G. Xiao, E.G. Kalayci, M. Roshchin, Semantically-enhanced rule-based diagnostics for industrial internet of things: The SDRL language and case study for Siemens trains and turbines, J. Web Semant. 56 (2019) 11–29.
[22] E. Kharlamov, D. Hovland, E. Jiménez-Ruiz, D. Lanti, H. Lie, C. Pinkel, M. Rezk, M.G. Skjæveland, E. Thorstensen, G. Xiao, D. Zheleznyakov, I. Horrocks, Ontology based access to exploration data at Statoil, in: ISWC, 2015, pp. 93–112.
[23] E. Kharlamov, N. Solomakhina, Ö.L. Özçep, D. Zheleznyakov, T. Hubauer, S. Lamparter, M. Roshchin, A. Soylu, S. Watson, How Semantic Technologies Can Enhance Data Access at Siemens Energy, in: ISWC, 2014, pp. 601–619.
[24] S. Elmer, F. Jrad, T. Liebig, A. ul Mehdi, M. Opitz, T. Stauß, D. Weidig, Ontologies and reasoning to capture product complexity in automation industry, in: ISWC (Posters & Demonstrations and Industry Tracks), 2017.
[25] J. Strötgen, T. Tran, A. Friedrich, D. Milchevski, F. Tomazic, A. Marusczyk, H. Adel, D. Stepanova, F. Hildebrand, E. Kharlamov, Towards the Bosch materials science knowledge base, in: ISWC (Posters & Demonstrations, Industry, and Outrageous Ideas), 2019, pp. 323–324.
[26] E.G. Kalayci, I. Grangel-González, F. Lösch, G. Xiao, A. ul Mehdi, E. Kharlamov, D. Calvanese, Semantic integration of Bosch manufacturing data using virtual knowledge graphs, in: ISWC, 2020, pp. 464–481.
[27] B. Zhou, Y. Svetashova, T. Pychynski, E. Kharlamov, Semantic ML for manufacturing monitoring at Bosch, in: ISWC (Demos/Industry), Vol. 2721, 2020, p. 398.
[28] B. Zhou, Machine Learning Methods for Product Quality Monitoring in Electric Resistance Welding (Ph.D. thesis), Karlsruhe Institute of Technology, Germany, 2021.
[29] ISO, 14327:2004: Resistance Welding – Procedures for Determining the Weldability Lobe for Resistance Spot, Projection and Seam Welding, International Organization for Standardization, Geneva, CH, 2004.
[30] DVS, 2902–3: Widerstandspunktschweißen von Stählen bis 3 mm Einzeldicke – Konstruktion und Berechnung, Deutscher Verband für Schweiß en und verwandte Verfahren e. V., Düsseldorf, DE, 2016.
[31] B. Zhou, Y. Svetashova, S. Byeon, T. Pychynski, R. Mikut, E. Kharlamov, Predicting quality of automated welding with machine learning and semantics: a Bosch case study, in: 29th ACM International Conference on Information and Knowledge Management, ACM, 2020, pp. 2933–2940.
[32] B. Zhou, Y. Svetashova, T. Pychynski, I. Baimuratov, A. Soylu, E. Kharlamov, SemFE: Facilitating ML pipeline development with semantics, in: 29th ACM International Conference on Information and Knowledge Management, ACM, 2020, pp. 3489–3492.

[33] Y. Svetashova, B. Zhou, S. Schmid, T. Pychynski, E. Kharlamov, SemML: Reusable ML models for condition monitoring in discrete manufacturing, in: ISWC (Demos/Industry), Vol. 2721, 2020, pp. 213–218.
[34] B. Zhou, T. Pychynski, M. Reischl, E. Kharlamov, R. Mikut, Machine learning with domain knowledge for predictive quality monitoring in resistance spot welding, J. Intell. Manuf. (2021) in press.
[35] DIN, Maintenance-Maintenance Terminology, Trilingual Version EN 13306:2017, 13306, 2018, p. 2017.
[36] J.N. Desai, S. Pandian, R.K. Vij, Big data analytics in upstream oil and gas industries for sustainable exploration and development: A review, Environ. Technol. Innov. (2020) 101186.
[37] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, From data mining to knowledge discovery in databases, AI Mag. 17 (3) (1996) 37.
[38] R. Mikut, M. Reischl, O. Burmeister, T. Loose, Data mining in medical time series, Biomed. Tech. 51 (2006).
[39] A. Soylu, Ó. Corcho, B. Elvesæter, C. Badenes-Olmedo, F.Y. Martínez, M. Kovacic, M. Posinkovic, I. Makgill, C. Taggart, E. Simperl, T.C. Lech, D. Roman, Enhancing public procurement in the European Union through constructing and exploiting an integrated knowledge graph, in: ISWC, 2020, pp. 430–446.
[40] S. Shalev-Shwartz, S. Ben-David, Understanding machine learning: From theory to algorithms.
[41] P.M. LaCasse, W. Otieno, F.P. Maturana, A survey of feature set reduction approaches for predictive analytics models in the connected manufacturing enterprise, Appl. Sci. 9 (5) (2019) 843.
[42] Y. Bengio, A. Courville, P. Vincent, Representation learning: A review and new perspectives, IEEE Trans. Pattern Analy. Mach. Intell. 35 (8) (2013) 1798–1828.
[43] A. Soylu, F. Mödritscher, F. Wild, P. De Causmaecker, P. Desmet, Mashups by orchestration and widget-based personal environments: Key Challenges, solution strategies, and an application, Program 46 (4) 383–428.
[44] C. Simon, Extensions to the semantic sensor network ontology, 2018, W3C Working Draft.
[45] A. Haller, K. Janowicz, S.J. Cox, M. Lefrançois, K. Taylor, D. Le Phuoc, J. Lieberman, R. García-Castro, R. Atkinson, C. Stadler, The modular ssn ontology: A joint w3c and ogc standard specifying the semantics of sensors, observations, sampling, and actuation, Semant. Web 10 (1) (2019) 9–32.
[46] M.G. Skjæveland, D.P. Lupp, L.H. Karlsen, H. Forssell, Practical ontology pattern instantiation, discovery, and maintenance with reasonable ontology templates, in: International Semantic Web Conference, Springer, 2018, pp. 477–494.
[47] P. Hitzler, M. Krötzsch, B. Parsia, P.F. Patel-Schneider, S. Rudolph, OWL 2 web ontology language primer, W3C Recomm. 27 (1) (2009) 123.
[48] B. Zhou, T. Pychynski, M. Reischl, R. Mikut, Comparison of machine learning approaches for time-series-based quality monitoring of resistance spot welding (RSW), Arch. Data Sci. Ser. A (Online First) 5 (1) (2018) 13.
[49] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[50] C. ISO, 9241-11.3.(1993) Part II: Guidance on specifying and measuring usability, Iso 9241 Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs).
[51] P. Ristoski, H. Paulheim, Semantic web in data mining and knowledge discovery: A comprehensive survey, J. Web Semant. 36 (2016) 1–22.
[52] F. Jaensch, A. Csiszar, C. Scheifele, A. Verl, Digital twins of manufacturing systems as a base for machine learning, in: 2018 25th International Conference on Mechatronics and Machine Vision in Practice, M2VIP, IEEE, 2018, pp. 1–6.
[53] S. Borgo, P. Leitão, The role of foundational ontologies in manufacturing domain applications, in: OTM, 2004.
[54] Z. Usman, R.I.M. Young, N. Chungoora, C. Palmer, K. Case, J. Harding, A manufacturing core concepts ontology for product lifecycle interoperability, in: International IFIP Working Conference on Enterprise Interoperability, Springer, 2011, pp. 5–18.
[55] S. Lemaignan, A. Siadat, J.-Y. Dantan, A. Semenenko, MASON: a proposal for an ontology of manufacturing domain, in: IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications, DIS'06, IEEE, 2006, pp. 195–200.
[56] S. Krima, R. Barbau, X. Fiorentini, R. Sudarsan, R.D. Sriram, OntoSTEP: OWL-DL Ontology for STEP, Tech. rep. 7561, NIST, 2009.
[57] X. Fiorentini, I. Gambino, V.-C. Liang, S. Rachuri, M. Mani, C.B. Nistir, C. Bock, C.M. Gutierrez, J.M. Turner, An Ontology for Assembly Representation, Tech. rep., Citeseer, 2007.
[58] D. Šormaz, A. Sarkar, SIMPM – Upper-level ontology for manufacturing process plan network generation, Robot. Comput.-Integr. Manuf. 55 (2019).
[59] E.G. Kalaycı, I.G. González, F. Lösch, G. Xiao, A. ul Mehdi, E. Kharlamov, D. Calvanese, Semantic integration of bosch manufacturing data using virtual knowledge graphs, in: ISWC, 2020.
[60] R. Hai, S. Geisler, C. Quix, Constance: An intelligent data lake system, SIGMOID'16.
[61] C. Quix, R. Hai, I. Vatov, GEMMS: A generic and extensible metadata management system for data lakes, in: CAiSE Forum, Vol. 129, 2016.
[62] S. Jupp, T. Burdett, D. Welter, S. Sarntivijai, H. Parkinson, J. Malone, Webulous and the webulous Google add-on-a web service and application for ontology building from templates, J. Biomed. Semant. 7 (2016).
[63] H. Dietze, T.Z. Berardini, R.E. Foulger, D.P. Hill, J. Lomax, D. Osumi-Sutherland, P. Roncaglia, C.J. Mungall, Termgenie-a web-application for pattern-based ontology class generation, J. Biomed. Semant. 5 (2014).
[64] Z. Xiang, J. Zheng, Y. Lin, Y. He, Ontorat: Automatic generation of new ontology terms, annotations, and axioms based on ontology design patterns, J. Biomed. Semant. 6 (1) (2015) 1–10.
[65] I. Boersch, U. Füssel, C. Gresch, C. Großmann, B. Hoffmann, Data mining in RSW, Int. J. Adv. Manuf. Technol. (2016) 1–15.
[66] K.-Y. Kim, F. Ahmed, Semantic weldability prediction with Rsw quality dataset and knowledge construction, Adv. Eng. Inf. 38 (2018) http://dx.doi.org/10.1016/j.aei.2018.05.006.
[67] A. Sumesh, K. Rameshkumar, K. Mohandas, R.S. Babu, Use of ML algorithms for weld quality monitoring using Acoustic signature, Procedia Comput. Sci. 50 (2015) 316–322.
[68] J. Yu, Quality estimation of resistance spot weld based on logistic regression analysis of welding power signal, Int. J. Precis. Eng. Manuf. 16 (13) (2015) 2655–2663.
[69] S. Saha, Z. Usman, W. Li, S. Jones, N. Shah, Core domain ontology for joining processes to consolidate welding standards, Robot. Comput.-Integr. Manuf. 59 (2019) 417–430.
[70] H. Cešpivová, J. Rauch, V. Svatek, M. Kejkula, M. Tomeckova, Roles of medical ontology in association mining crisp-DM cycle, in: ECML/PKDD04, 2004.
[71] P. Ding, S. Jun-yi, Z. Mu-xin, Incorporating domain knowledge into data mining process: An Ontology based framework, Wuhan Univ. J. Nat. Sci. 11 (2006) 165–169.
[72] S. Evert, P. Heinrich, K. Henselmann, U. Rabenstein, E. Scherr, M. Schmitt, L. Schröder, Combining machine learning and Semantic features in the classification of corporate disclosures, J. Log. Lang. Inf. 28 (2) (2019) 309–330.
[73] J.M. Perea-Ortega, M.T. Martín-Valdivia, L.A. Ureña López, E. Martínez-Cámara, Improving polarity classification of bilingual parallel corpora combining machine learning and semantic orientation approaches, J. Am. Soc. Inf. Sci. Technol. 64 (9) (2013) 1864–1877.
[74] A. Seeliger, M. Pfaff, H. Krcmar, Semantic web technologies for explainable machine learning models: A literature review, in: PROFILES/SEMEX@ ISWC 2465, 2019, pp. 1–16.
[75] M.K. Sarker, N. Xie, D. Doran, M. Raymer, P. Hitzler, Explaining trained neural networks with semantic web technologies: First steps, 2017, arXiv preprint arXiv:1710.04324.
[76] P. Wang, Q. Wu, C. Shen, A.v.d. Hengel, A. Dick, Explicit knowledge-based reasoning for visual question answering, 2015, arXiv preprint arXiv:1511.02570.
[77] M. Batet, A. Valls, K. Gibert, Performance of Ontology-based Semantic similarities in clustering, in: International Conference on Artificial Intelligence and Soft Computing, Springer, 2010, pp. 281–288.
[78] I. Tiddi, M. d'Aquin, E. Motta, Dedalo: Looking for clusters explanations in a labyrinth of linked data, in: European Semantic Web Conference, Springer, 2014, pp. 333–348.
[79] O.Z. Khan, P. Poupart, J.P. Black, Explaining recommendations generated by MDPs, in: ExaCt, Citeseer, 2008, pp. 13–24.
[80] Y. Shi, G. Cheng, T. Tran, J. Tang, E. Kharlamov, Keyword-based knowledge graph exploration based on quadratic group Steiner trees, in: IJCAI, 2021, pp. 1555–1562.
[81] Y. Shi, G. Cheng, T. Tran, E. Kharlamov, Y. Shen, Efficient computation of semantically cohesive subgraphs for keyword-based knowledge graph exploration, in: WWW, 2021, pp. 1410–1421.
[82] Y. Shi, G. Cheng, E. Kharlamov, Keyword search over knowledge graphs via static and dynamic hub labelings, in: WWW, 2020, pp. 235–245.
[83] E. Kharlamov, L. Giacomelli, E. Sherkhonov, B.C. Grau, E.V. Kostylev, I. Horrocks, SemFacet: Making hard faceted search easier, in: CIKM, 2017, pp. 2475–2478.
[84] E. Sherkhonov, B.C. Grau, E. Kharlamov, E.V. Kostylev, Semantic faceted search with aggregation and recursion, in: ISWC, 2017, pp. 594–610.
[85] M. Arenas, B.C. Grau, E. Kharlamov, S. Marciuska, D. Zheleznyakov, Faceted search over RDF-based knowledge graphs, J. Web Semant. 37–38 (2016) 55–74.
[86] M. Arenas, B.C. Grau, E. Kharlamov, S. Marciuska, D. Zheleznyakov, Faceted search over ontology-enhanced RDF data, in: CIKM, 2014, pp. 939–948.
[87] J. Li, G. Cheng, Q. Liu, W. Zhang, E. Kharlamov, K. Gunaratna, H. Chen, Neural entity summarization with joint encoding and weak supervision, in: IJCAI, 2020, pp. 1644–1650.
[88] Q. Liu, Y. Chen, G. Cheng, E. Kharlamov, J. Li, Y. Qu, Entity summarization with user feedback, in: ESWC, 2020, pp. 376–392.

[89] G. Cheng, K. Gunaratna, E. Kharlamov, Entity summarization in knowledge graphs: Algorithms, evaluation, and applications, in: WWWW, 2020, pp. 301–302.
[90] S. Li, Z. Huang, G. Cheng, E. Kharlamov, K. Gunaratna, Enriching documents with compact, representative, relevant knowledge graphs, in: IJCAI, 2020, pp. 1748–1754.
[91] Z. Huang, S. Li, G. Cheng, E. Kharlamov, Y. Qu, MiCRon: Making sense of news via relationship subgraphs, in: CIKM, 2019, pp. 2901–2904.
[92] X. Wang, G. Cheng, E. Kharlamov, Towards multi-facet snippets for dataset search, in: Joint Proceedings of the 6th International Workshop on Dataset PROFlLing and Search & the 1st Workshop on Semantic Explainability Co-Located with the 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 27, 2019, 2019, pp. 1–6.
[93] X. Wang, J. Chen, S. Li, G. Cheng, J.Z. Pan, E. Kharlamov, Y. Qu, A framework for evaluating snippet generation for dataset search, in: ISWC, 2019, pp. 680–697.
[94] J. Chen, X. Wang, G. Cheng, E. Kharlamov, Y. Qu, Towards more usable dataset search: From query characterization to snippet generation, in: CIKM, 2019, pp. 2445–2448.
[95] X. Wang, G. Cheng, J.Z. Pan, E. Kharlamov, Y. Qu, BANDAR: Benchmarking snippet generation algorithms for (RDF) dataset search, IEEE Trans Knowl. Data Eng. (2021) http://dx.doi.org/10.1109/TKDE.2021.3095309.
[96] A. Soylu, M. Giese, E. Jiménez-Ruiz, E. Kharlamov, D. Zheleznyakov, I. Horrocks, Ontology-based end-user visual query formulation: Why, what, who, how, and which? Univers. Access Inf. Soc. 16 (2) (2017) 435–467.
[97] X. Wang, G. Cheng, T. Lin, J. Xu, J.Z. Pan, E. Kharlamov, Y. Qu, PCSG: pattern-coverage snippet generation for RDF datasets, in: ISWC, 2020.
[98] C. Pinkel, C. Binnig, E. Jiménez-Ruiz, E. Kharlamov, W. May, A. Nikolov, A.S. Bastinos, M.G. Skjæveland, A. Solimando, M. Taheriyan, C. Heupel, I. Horrocks, RODI: Benchmarking relational-to-ontology mapping generation quality, Semant. Web 9 (1) (2018) 25–52.
[99] E. Jiménez-Ruiz, E. Kharlamov, D. Zheleznyakov, I. Horrocks, C. Pinkel, M.G. Skjæveland, E. Thorstensen, J. Mora, BootOX: Practical mapping of RDBs to OWL 2, in: ISWC, 2015, pp. 113–132.