Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 42 (2020) 375–382

Procedia MANUFACTURING

www.elsevier.com/locate/procedia

International Conference on Industry 4.0 and Smart Manufacturing (ISM 2019)

Metadata management in a big data infrastructure

Roxana-Maria Holom[a,*], Katharina Rafetseder[a], Stefanie Kritzinger[a], Harald Sehrschön[b]

[a]*RISC Software GmbH, Softwarepark 35, Hagenberg 4232, Austria*
[b]*Fill Gesellschaft m.b.H., Fillstrasse 1, Gurten 4942, Austria*

Abstract

The adoption of the Internet of Things (IoT) in industry provides the opportunity to gather valuable data. Nevertheless, this amount of data must be analyzed to identify patterns in the data, model behaviors of equipment and to enable prediction. Although big data found its initiation already some years ago, there are still many challenges to be solved, e.g. metadata representation and management are still a research topic. The big data architecture of the RISC data analytics framework relies on the combination of big data technologies with semantic approaches, to process and store large volumes of data from heterogeneous sources, provided by FILL, which is a key machine tool provider. The proposed architecture is capable of handling sensor data using big data technologies such as Spark on Hadoop, InfluxDB and Elasticsearch. The metadata representation and management approach is adopted in order to define the structure and the relations (i.e., the connections) between the various data sources provided by the sensors and logging information system. On the other hand, using a metadata approach in our big data environment enhances RISC data analytics framework by making it generic, reusable and responsive in case of changes, thus keeping the data lakes up-to-date and ensuring the validity of the analytics results. The work presented here is part of an ongoing project (BOOST 4.0) currently addressed under the EU H2020 program.

Keywords: big data; metadata; data harmonization; linked data; semantics; machine learning.

1. Introduction

In the context of the European project called Big Data Value Spaces for COmpetitiveness of European COnnected Smart FacTories 4.0 (BOOST 4.0) [19, 2] RISC Software GmbH (RISC) and Fill Gesellschaft m.b.H. (FILL) together with other partners try to achieve a better understanding of the machinery by detecting cause-and-effect relationships in the related stored data. Thus, understanding the data, finding patterns in the data, generating prediction models and checking machine data against machine specification have been identified as main project goals.

The evolvement of cloud and cluster infrastructures together with the adoption of the Internet of Things (IoT) in industry enabled the rise of big data and concurrently reinvented the idea of Artificial Intelligence (AI). Most organizations today see their data as a big opportunity to gather insights into their processes, tools, machines, materials, etc. After gathering data, one starts to think about exploiting data in order to generate useful results that can be further used in the big data engineering and analysis process.

To address big data challenges as data and infrastructure management, i.e., to be able to do something useful with large amounts of data, an innovative big data strategy based on metadata representation [4] is indispensable. Most of the IoT and big data platform approaches focus on operations and not on the engineering process of such production systems. Closing this loop leads to the next evolutionary step in the engineering methodologies like model based engineering and model based systems engineering. RISC data analytics framework implements the previously mentioned big data strategy by applying lossless and highest standardized data interoperability between engineering tools, processes and domains at different levels: process, technical, syntax and semantic level [31].

The data sets generated by sensors can be analyzed using methods from data mining, such as machine learning, which can be used to derive information about correlations, existing patterns and anomalies in the data pool.

This paper describes an approach for metadata representation and management in a big data environment, based on a real

* Corresponding author. Tel.: +43 664 2482603.
E-mail address: roxana.holom@risc-software.at (Roxana-Maria Holom).

2351-9789
10.1016/j.promfg.2020.02.060

application example. Initially, we shortly describe the goals and objectives of RISC and FILL together with the other partners inside BOOST 4.0 and conclude with the identified challenges related to the analysis objectives.

2. FILL pilot in BOOST 4.0

BOOST 4.0 project addresses the need for development of large scale experimentation and demonstration of data-driven "connected smart" Factories 4.0 by demonstrating in a measurable and replicable way, a shared data-driven Factory 4.0 model through 10 lighthouse factories. One of these 10 factories is represented by the FILL Pilot, who addresses this objective from the proposal: leverage smart integration of industry 4.0, real-time architectures, big data predictive analytics and data/knowledge protection with Industrial Data Spaces. Activities under this objective are: design and deliver a scalable holistic analytical framework - data mining, machine learning, information visualization, correlation analysis and user interaction methods for heterogeneous for big industrial data.

2.1. Short pilot overview

The FILL pilot is about a lot-size-one machine tool circular engineering in a factory 4.0. FILL as key machine tool provider delivers in the order of 100 production systems per year to around 50 customers worldwide. 90% of FILL machines and systems are exported to automotive, aerospace, sports, construction and housing and renewable energy sectors. Zero defect production demand that machines cost-effectively flexibly adapt to optimum production configurations. Therefore, machine tools are increasingly customizable (lot-size 1 production scheme). Rigid engineering processes designed for mass production are not able to optimize smart connected machine tool lot-size 1 engineering and fail to incorporate external operational data to optimize machine self-configuration and adaptation features. The FILL trial primarily serves the engineering process of the machine builder. It allows for a better understanding of machinery by detecting cause-and-effect relationships due to anomalies and patterns.

FILL's system *Machine Workflow* collects data from the production process of the machine, stores them in database systems and makes them available for further evaluation with regard to the produced product. Currently, FILL provides us historical batch data from InfluxDB and Elasticsearch. In an initial step we hope to get more closely insights about the machines and their conditions during operation through data exploration and analysis and, in a subsequent step, to establish a pattern and anomaly detection framework based on machine learning algorithms.

2.2. Challenges for the data analysis

It is well known that in some cases the schema-less environment offered by big data ecosystems is seen as an advantage, because of the agile nature and flexibility of the data development. But, interpreting the data available in a big data system does not receive much support from this thinking. As mentioned in [25], the need for metadata management support in big data environments was proved by use cases that require validity of analytics results or when one wants to scale from answering a small narrow business question to a large-scale analysis environment. We systematically analyzed our use cases in this project and from the many challenges big data poses [24] we identified the following:

- machine learning algorithms are not that flexible and their use in an analysis that spans over long time periods represents a big challenge;
- integration of machine learning algorithms in a large-scale analysis environment can be quite complicated;
- many analysis conditions used in machine learning algorithms are being hardcoded;
- ensuring the validity of the analysis;
- frequent evolution of data sources;
- sharing of algorithms.

After researching for similar problems and their related solutions, we understood that defining metadata for big data helps us in overcoming the previously mentioned challenges.

3. Related Work

Taking it incrementally, we started by searching for solutions to represent our data via suitable schemata, but this is not straightforward in big data context. Therefore, the next thought was to represent and manage metadata, i.e., the syntax and the semantics of the data. By semantics we limited ourselves to represent the relations between the data. Starting with these ideas in mind, we discovered that the interest in using metadata in big data environments has increased over the last few years. Local different solutions were developed by big companies or research groups. Open II [23] addresses a more generic approach by creating and integrating tools for metadata in big data infrastructures. Though a great idea, it looks like this work was no longer continued, because we couldn't find how we could apply it to our specific data sources. LinkedIn tried to overcome some of the challenges mentioned before by engineering a data model based on combining Apache Avro [26] and Pegasus [12, 13]. Differently, our data pipeline is offline, so using Avro in our case will not require the integration of Kafka [28, 8].

Another approach to deal with the non-uniformity of big data is related to graph databases, whose use significantly increased over the last years [5]. Graph databases are considered schema-less, which make them suitable for big data and therefore many companies have turned to use them [3, 9, 14]. Although already many tools integrate graph databases and ontologies [15] with Spark, these can be explored only using RDF-specific query languages, like SPARQL [32] or Streaming and Temporal ontology Access with a Reasoning-based Query Language (STARQL)-SPARK query engine [18, 21]. Thus, knowledge and experience with Web Ontology Language (OWL) and Resource Description Framework (RDF) are required.

We want to develop a different approach, which uses some features offered by ontologies for the metadata management, but in the backend, does not make use of the traditional implementation based on OWL and RDF, and therefore supporting the use of the default querying language in Spark. This way we want to reduce the challenges posed to our developers (which are not having a background in semantics) by specific query languages, like SPARQL or STARQL. Of course, on the one hand, as a consequence the benefits proposed by ontologies will be reduced, but, on the other hand, by integrating Avro and Semantic Annotations for Linked Avro Data (SALAD) schemas, we will still be able to design the data model with the corresponding metadata and specify validation conditions.

4. RISC data analytics framework

The main tasks of RISC in the pilot factory led by FILL are focusing on the selection of appropriate machine learning and data analysis algorithms and their algorithmic optimizations for the given problem statements, i.e., find the appropriate methods that are suitable for very large data and that have the potential for parallel implementation.

4.1. Motivation

After analyzing the big data use cases we are dealing with, significant aspects for the project have been identified, namely, the analysis should work over long periods of time, data should be consolidated (some information is derived from another and this should be kept clear), combining data should not break the analytic algorithms and data discovery should be facilitated. For now, we do an historical analysis, but the data is relatively new (approximately one year and during this time the data has changed several times). Therefore, the algorithms should be developed to analyze in an initial phase the historical data we have, but after being integrated in the FILL ecosystem, they should succeed in analyzing near to real time data. Thus, reusability is very important. Another aspect, which requires flexible and reusable algorithms is the lack of historical bad data (data that represents failures). In this case, the algorithms should learn over time from the changing data sets.

To accomplish the tasks defined in the project, we understood that we have to enhance our data analytics framework in the direction of big data fabric [33]. One of the biggest challenges identified was dealing with disparate data sources, which provide high velocity sensor data, on one side, and log information (alarm messages), on the other side. Additionally, another challenge represents linking the data provided by the different data sources. Therefore, democratizing data by representing related metadata is indispensable in order to minimize the effort spent for ingesting, integrating, curating the data [33], i.e., to reduce the complexity of the data analysis process. Data analysis is a cumbersome process of reviewing, cleansing, transforming and modeling data with the aim of discovering useful information, drawing conclusions and supporting decision-making.

To succeed with our objectives, we need to design a centralize pipeline. This requires to integrate a metadata representation approach in our big data infrastructure. All data is ingested into our data storage layer, which builds on top of a central repository (i.e., reader and writer always use the same schema) containing all metadata schemas. Relational mapping is not a trivial task in big data, therefore special handling is required. Model information of the machines is designed in the form of ontologies and registered in a generic database system.

4.2. Structuring big data using ontologies

Semantic technology [30] was initially used in the Semantic Web context in order to allow the meaning of associations between information (building relationships between various formats of data) to be known and processed at execution time. Therefore, metadata is a fundamental component of the Semantic Web, together with the main standards on which it builds on: the RDF, SPARQL and OWL. Another very important layer in the Semantic Web's architecture is the "Ontology vocabulary", which can be translated into metadata vocabularies and therefore considered to be the Semantic Web's central metadata artery [10].

The term ontology originally comes from the field of theoretical philosophy. In computer science, the use of ontologies is a common methodology for modelling knowledge digitally and formally. Hence, it is a formal description of complex facts and expertise (to conserve and share it, but also to machine it (further) to process and expand [22, 11, 6]). While the use of ontologies has long been limited to the scientific-academic field (especially computer science biomedicine, biotechnology and medical informatics), in recent years, more and more applications in the area of big data, data integration, data analysis and industry 4.0 have been found [7].

Ontologies use semantic search technology in order to discover meaningful information from structured and unstructured data. They also help addressing the challenges that data integration presents: data sourcing, data connectivity and data migration. Another benefit is the ability to overcome the lack of flexibility and the significant delay time that data integration cannot cope with during changes in data types, data sources, or datasets. A big topic for ontologies is the connection of data from different data sources and their interpretations. The data agility, data-first approach - data building (data first, schema / structure later) - and the step-by-step data integration process provide a foundation for a new way to manage enterprise data collections and big data [17, 16].

Behind the tasks of the selection of the most appropriate machine learning algorithms, data analysis algorithms and their optimizations, a more hidden goal of our team is to create a big data analysis environment that enables ontology modeling for metadata management and metadata representation, together with data exploration and analysis for big data lakes.

One of the reasons for the decision of employing an ontology-based approach that relinquishes their traditional use based on RDF and SPARQL, was lowering the fundamental barriers to adoption of semantic technologies. However, this

approach still enables some of the benefits of the knowledge graph capabilities by allowing us to keep the initial database infrastructure. Another reason to base our implementation on the W3C Semantic Web principles is to enable relations between concepts in Structured Query Language (SQL). Moreover, we can do complex analytics using the same tools we currently use to access and query the databases mentioned in Fig. 1 and subsection 4.3 and in addition, this concept facilitates sophisticated query of big data without requiring high knowledge in semantic technology.

RISC data analysis framework employs Spark SQL, which allows relational queries expressed in SQL – the most widely known database language, to eliminate the technological barriers to entry for using knowledge graphs. Therefore, together with the integration of Avro and Salad, it enables modeling of data as connected, context-enriched concepts with inference, while being queryable in standard SQL, to transform data into connected data that is seamlessly accessible to our machine learning algorithms. Modeling of concepts extends schema and table definitions with abstractions, in form of metadata. Hence, our framework facilitates standardization of data meaning across our team and makes data understandable to both users and machines. It also enables data engineers and scientists to be highly productive doing data preprocessing, feature engineering, data analysis and building accurate and flexible machine learning models.

In the next subsections, the focus lies on describing the big data architecture we build on and first ideas on representing the metadata with Avro and Salad. As future work, we plan to implement the metadata management by extending our in-house ontology-based tool, to integrate it with the metadata representation technologies and, in the last stage of the project, incorporate the data exploration and visualization approaches and the appropriate machine learning algorithms.

4.3. Big data architecture

The big data architecture of the RISC data analytics framework is designed as pictured in Fig. 1 and relies on the open source software collection Apache Hadoop [27] for data storage and computation. Running on a cluster of computers, Apache Hadoop provides a software framework for distributed storage and processing of big data using the MapReduce programming model. To process data in parallel, Hadoop sends packaged code to each node and takes advantage of data locality, meaning that each node only manipulates the data it has access to. We rely on Apache Spark [29] for cluster computing, because it provides high-level APIs in various languages (Java, Scala, Python and R) that perfectly integrate with machine learning libraries. Spark keeps track of the data that each of the operators produces, and enables applications to reliably store this data in memory. This is the key to Spark's performance, as it allows applications to avoid costly disk accesses.

Spark programs can be submitted from outside to the cluster, for example from within the web-based IPython interpreter Jupyter or directly from an IDE (e.g. by using Livy-Sessions). This way, after the cluster has done some heavy lifting, results

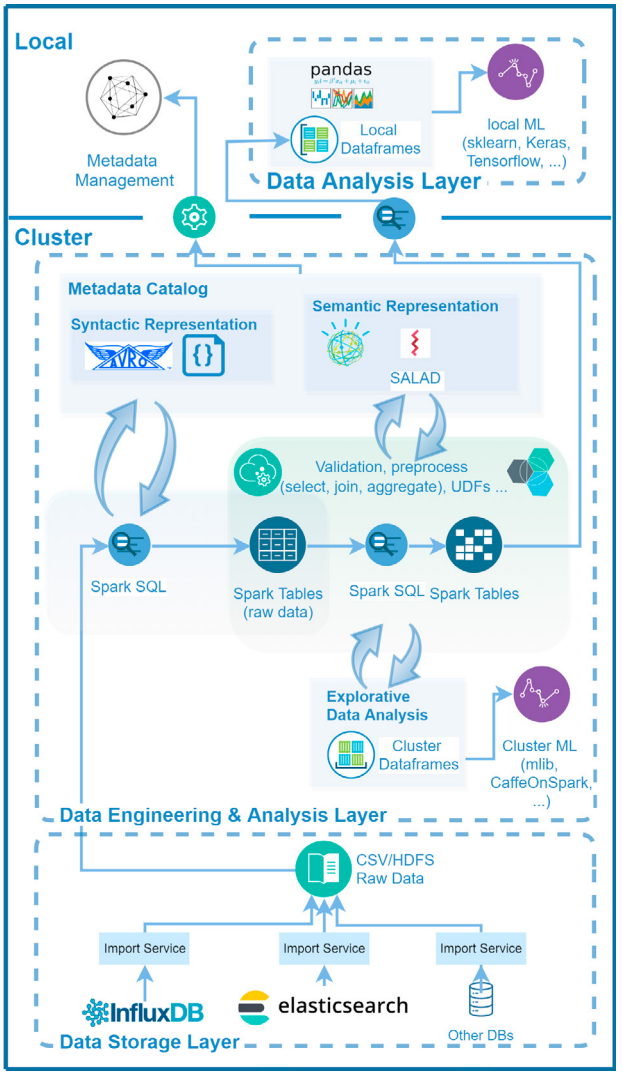

Fig. 1. RISC data analytics framework - Big Data architecture

can be plotted immediately using libraries such as *matplotlib* or *plotly*.

As it can be seen Fig. 1 the architecture is split into two components: the cluster, which is the main component, and the local component. Most of the data engineering is happening on the cluster side, where two layers were designed: the data storage and the data engineering and analysis layer.

In a first step, the data which arises from various data sources is being ingested in the corresponding databases in the data storage layer (big data batches - sensor data of the production process or machine operation - are stored in an InfluxDB and log information is stored in Elasticsearch). Afterwards, the data sets are interpreted in the appropriate way and imported to comma-separated values (CSV) files, which are stored on Hadoop Distributed File System (HDFS), in the data engineering and analysis layer. At this initial stage, only simple data

validation and correction happens. In order to enhance the flexibility and applicability of the system, additional input data connectors were implemented. At syntax level, the structure of the imported data was defined by representing its related metadata information through Avro schemas. In a further step, the data is being validated, cleaned, correspondingly transformed and stored into Spark tables. Using Spark SQL together with Livy, we can rapidly conduct some first data explorations in order to gain initial insights about the data.

A central aspect of this project is the involvement of the experts in data analysis and interpretation. After seeing the first data explorations and consulting the domain experts, a semantic representation of the metadata is designed, with a focus on connecting the information from the multiple sources, i.e., linking the data. By implementing the relations between the different data sources at a different level, the complexity of the queries is reduced (i.e., less JOINs). Therefore, the data analysis process becomes simpler and more intuitive.

For the metadata management task, an in-house ontology-based research infrastructure for domain-expert-driven knowledge discovery is being used. The main principle behind the system is to put the domain experts with their knowledge and experience in the center of the knowledge discovery process. Therefore, the metadata information of the linked data and model information of the machine (i.e., one of the machines produced by FILL) are being represented as a domain ontology in a generic database system (different from a traditional ontology system based on OWL and RDF). This tool is mainly based upon Java enterprise technology and uses Maria DB as data storage system. The system is also able to access data from external systems, by either importing it into its internal data storage or accessing the external data at runtime. We enhanced the ontology-based tool in order to able to integrate it as a backend component into our data analytics framework for managing the metadata representation, especially for the semantic information necessary for linking the data. Furthermore, the elaborated meta-information from the domain ontology is used to actively support the data scientists in complex data science tasks, such as data preparation, data selection and filtering, data cleaning and plausibilization, data exploration and analytics.

As mentioned before, we use Avro as intermediate format to define data schemas in JavaScript Object Notation (JSON) for representing the syntactic metadata. The implementation is straightforward, but it is not rich enough to facilitate the representation of data relations. Therefore, we have researched for a more powerful metadata representation approach and discovered Salad [1]. Salad is a schema language for describing structured linked data in JSON or YAML (a recursive acronym for "YAML Ain't Markup Language") documents. By declaring a Salad schema, rules for preprocessing, structural validation, and link checking for documents can be defined. Salad fills in the gap between the record oriented data modeling supported by Apache Avro and the Semantic Web, by extending Avro with features for rich data.

Data aggregation and record linkage is followed by data processing and query steps. At this stage, explorative data analysis is conducted to properly prepare the data for time series analysis (as it is in our case). From here on, there are two possibilities: to reduce the data and load it locally into pandas data frames that will feed the machine learning models or train the machine learning models on the cluster using libraries like *mlib*, *CaffeOnSpark*, or *H2O (Sparkling Water)*.

4.4. Representing metadata

The AVRO schema plays a key role in defining the syntax of the data being processed and can be used to validate data as well as for automated data processing operations. Some of the advantages of Avro that convinced us to use it for representing data information are that it integrates with Hadoop, schema changes are handled transparently and it offers programmatic compatibility model.

While the AVRO schema only contains syntactic information, it can be used to convey information, which is required for automated data processing, such as the resolution of the `Timestamp` information as in the example in Listing 1. While it may be obvious to a human reader, whether the long value of `time` represents milli- or microseconds, that information is crucial for the correct automated conversion to other datatypes such a Spark `timestamp`. Such information can be attributed to the struct fields within the schema.

```
{
  "type" : "record",
  "name" : "OperatingState",
  "namespace" : "lfd",
  "fields" : [
    {
      "name" : "time",
      "type" :
      {
        "type" : "long",
        "logicalType" : "timestamp-micros"
      }
    },
    {
      "name" : "active",
      "type" : "boolean"
    },
    {
      "name" : "name",
      "type" : "string"
    }
  ]
}
```

Listing 1. An example of an Avro schema

A downside of Avro is that it does not support the definition of direct relationships. It turns out that Avro does natively not facilitate a way of linking data across different types. The only option would be to nest the data for one entity within another entity data. However, this is not a flexible solution [20]. For this reason, Salad schema was created, i.e., a schema language that supports linked data through annotations, which describe the linked data interpretation of the content. We will define relations between data and describe rules for preprocessing and structural validation inside Salad schemas. Beside offering rich

data modeling with inheritance, Salad looks promising to us, because it supports object identifiers, object references, documentation generation and code generation (see Listing 2). In this paper we provide some Salad schemas examples, in which we describe YAML structured linked data documents.

```
$ schema-salad-tool --codegen=python
alarm_schema.yml > alarm_schema.py
```
Listing 2. Generate Python classes described by the Salad schema

Listing 3 is our starting point, in which we define the top entity, meaning the log file that contains all the alarms stored in Elasticsearch. Therefore, we have to continue by defining a schema for the `Alarm` record. As it can be seen in Listing 4, the fields which are known are being defined.

```
- name: LogInfo
  doc: |
    A collection of alarms. The
    'documentRoot' field indicates it is a
    valid starting point for a document.
    The 'logInfo' field will
    validate subtypes of 'Alarm'.
  type: record
  documentRoot: true
  fields:
    logInfo:
      type:
        type: array
        items: Alarm
```
Listing 3. Salad schema for the alarms saved inside Elasticsearch

```
- name: Alarm
  doc: |
    The base type for an alarm. This is an
    abstract type, so it can't be used
    directly, but can be used to define
    other types.
  type: record
  abstract: true
  fields:
    number: integer
    text: string
    priority:
      type:
        type: enum
        symbols:
          - message
          - warning
          - failure
    source: string
    manumber: string
```
Listing 4. Salad schema for an alarm

We have decided to extend the `Alarm` record based on the values of the `state` field, in order to be more flexible when validating the pairwise alarms (in a perfect scenario, an alarm has an entry in the database for the time when it was triggered and one for the time when it was solved). This is possible in Salad schema by using the `extends` keyword.

```
- name: AlarmTriggered
  doc: |
    An alarm that is triggered.
  type: record
  extends: Alarm
  fields:
    state:
      type:
        type: enum
        symbols:
          - 21
      jsonldPredicate: '#state'
    start: long
```
Listing 5. Salad schema that extends Alarm record

```
- name: AlarmSolved
  doc: |
    An alarm that is solved.
  type: record
  extends: Alarm
  fields:
    state:
      type:
        type: enum
        symbols:
          - 20
      jsonldPredicate: '#state'
    ends: long
```
Listing 6. Salad schema that extends Alarm record

After defining the salad schemas one can validate the schema against the related document by using the following command:

```
$ schema-salad-tool alarm_schema.yml
alarm.yml
Document ‘alarm.yml‘ is valid
```
Listing 7. Command to validate a Salad schema

5. Future Work

As mentioned before, the work presented in this paper is part of an European project that runs till the end of the year 2020, i.e., it is an ongoing work. Thus, the activities mentioned below will be conducted during the remaining of this year and next year.

Following the ideas presented in section 4, we will fine-tune the representation of the metadata for the data sources described under the data storage layer. Afterwards, the in-house ontology-based tool will be extended such that it will enable the management of the metadata represented in form of Avro and Salad schemas, i.e., the resulting schemas will be integrated. In a further step, the schemas will be broaden in order to include metadata that will be used by the data analysis algorithms. Accomplishing these tasks, we will continue exploring the data and will be able to start with the more closely related machine learning activities, like data segregation (feature engineering), model selection, model training, model evaluation and validation, model deployment and model integration in the large-scale data analysis system of FILL.

6. Conclusion

By representing the metadata information, we also move away from hardcoding essential information for the data analysis step inside the code. Hence, we revolutionize RISC data analytics framework by making it generic, reusable and responsive in case of changes. Keeping the data lake up-to-date is another important issue in big data infrastructures, because refreshing the content of a data lake is a complex process. Therefore, handling the integration of data in a data lake in an almost automatic way is a key feature of our framework.

The integration and synchronization of different data sources combining big data technologies with a semantic approach will improve the informative value of the analysis output, the quality of the developed models and will increase the model diversity.

Acknowledgements

The work described in this paper is supported by the BOOST 4.0 project, which has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 780732.

References

[1] Amstutz, P., . Semantic annotations for Linked Avro Data (SALAD). URL: https://www.commonwl.org/v1.0/SchemaSalad.html.

[2] BOOST 4.0 Project Partners, . Boost 4.0 - big data for factories. URL: https://boost40.eu/.

[3] Cambridge Semantics, . A semantic layer for hadoop. URL: https://info.cambridgesemantics.com/semantic-layer-for-hadoop?hsCtaTracking=ce9e1e0a-7848-454f-bd4d-f8e154f005e3%7C8d1aea90-3e2b-46bc-b483-21a26830e0a1.

[4] Ceravolo, P., Azzini, A., Angelini, M., Catarci, T., Cudré-Mauroux, P., Damiani, E., Mazak, A., Van Keulen, M., Jarrar, M., Santucci, G., Sattler, K.U., Scannapieco, M., Wimmer, M., Wrembel, R., Zaraket, F., 2018. Big data semantics. Journal on Data Semantics 7, 65–85. URL: https://doi.org/10.1007/s13740-018-0086-2, doi:10.1007/s13740-018-0086-2.

[5] Chalk, S.J., 2016. Scidata: a data model and ontology for semantic representation of scientific data. J. Cheminformatics 8, 54:1–54:24. URL: https://doi.org/10.1186/s13321-016-0168-9, doi:10.1186/s13321-016-0168-9.

[6] Chandrasekaran, B., Josephson, J.R., Benjamins, V.R., 1999. What are ontologies, and why do we need them? IEEE Intelligent Systems 14, 20–26. URL: https://doi.org/10.1109/5254.747902, doi:10.1109/5254.747902.

[7] Cheng, C., Guelfirat, T., Messinger, C., Schmitt, J.O., Schnelte, M., Weber, P., 2015. Semantic degrees for industrie 4.0 engineering: deciding on the degree of semantic formalization to select appropriate technologies, in: Nitto, E.D., Harman, M., Heymans, P. (Eds.), Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, Bergamo, Italy, August 30 - September 4, 2015, ACM. pp. 1010–1013. URL: https://doi.org/10.1145/2786805.2804434, doi:10.1145/2786805.2804434.

[8] Confluent Inc., . Camus. URL: https://docs.confluent.io/2.0.0/camus/docs/intro.html.

[9] Ermilov, I., Lehmann, J., Sejdiu, G., Bühmann, L., Westphal, P., Stadler, C., Bin, S., Chakraborty, N., Petzka, H., Saleem, M., Ngomo, A.N., Jabeen, H., 2017. The tale of sansa spark, in: Nikitina, N., Song, D., Fokoue, A., Haase, P. (Eds.), Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017., CEUR-WS.org. URL: http://ceur-ws.org/Vol-1963/paper552.pdf.

[10] Greenberg, J., Sutton, S., Campbell, D.G., 2003. Metadata: A fundamental component of the semantic web. Bulletin of the American Society for Information Science and Technology 29, 16–18. URL: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/bult.282, doi:10.1002/bult.282.

[11] Gruber, T.R., 1993. A translation approach to portable ontology specifications. Knowledge Acquisition 5, 199 – 220. URL: http://www.sciencedirect.com/science/article/pii/S1042814383710083, doi:https://doi.org/10.1006/knac.1993.1008.

[12] Hartman, J., . Parent-child relationships and you. URL: https://www.linkedin.com/pulse/parent-child-relationships-you-joshua-hartman/.

[13] Kreps, J., . Building linkedin's real-time data pipeline. URL: http://docs.huihoo.com/apache/kafka/Building-LinkedIn-Real-time-Data-Pipeline.pptx.

[14] Lehmann, J., Sejdiu, G., Bühmann, L., Westphal, P., Stadler, C., Ermilov, I., Bin, S., Chakraborty, N., Saleem, M., Ngomo, A.N., Jabeen, H., 2017. Distributed semantic analytics using the SANSA stack, in: International Semantic Web Conference (2), Springer. pp. 147–155.

[15] Nadal, S., Herrero, V., Romero, O., Abelló, A., Franch, X., Vansummeren, S., Valerio, D., 2017. A software reference architecture for semantic-aware big data systems. Information & Software Technology 90, 75–92.

[16] Ontology Systems, a. Ontology, big data and the logical data warehouse. URL: https://www.exfo.com/en/ontology/.

[17] Ontology Systems, b. Rethinking data integration in a post-google world. URL: https://www.exfo.com/en/ontology/.

[18] Özçep, Ö.L., Möller, R., Neuenstadt, C., 2014. A stream-temporal query language for ontology based data access, in: Description Logics, CEUR-WS.org. pp. 696–708.

[19] Project Partners of BOOST 4.0, . Big data value spaces for competitiveness of european connected smart factories 4.0 (boost 4.0). URL: https://cordis.europa.eu/project/rcn/213926/factsheet/en.

[20] Sarwar, S., . Schema salad in aether. URL: https://confluence.ehealthafrica.org/display/AET/Schema+Salad+in+Aether.

[21] Schiff, S., Möller, R., Özçep, Ö.L., 2019. Ontology-based data access to big data. OJDB 6, 21–32.

[22] Schmidhuber, J., 2015. Deep learning in neural networks: An overview. Neural Networks 61, 85–117.

[23] Seligman, L., Mork, P., Halevy, A.Y., Smith, K.P., Carey, M.J., Chen, K., Wolf, C., Madhavan, J., Kannan, A., Burdick, D., 2010. Openii: an open source information integration toolkit, in: Elmagarmid, A.K., Agrawal, D. (Eds.), Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010, ACM. pp. 1057–1060. URL: https://doi.org/10.1145/1807167.1807285, doi:10.1145/1807167.1807285.

[24] Sivarajah, U., Kamal, M.M., Irani, Z., Weerakkody, V., 2017. Critical analysis of big data challenges and analytical methods. Journal of Business Research 70, 263 – 286. URL: http://www.sciencedirect.com/science/article/pii/S014829631630488X, doi:https://doi.org/10.1016/j.jbusres.2016.08.001.

[25] Smith, K.P., Seligman, L.J., Rosenthal, A., Kurcz, C., Greer, M., Macheret, C., Sexton, M., Eckstein, A., 2014. "big metadata": The need for principled metadata management in big data ecosystems, in: Katsifodimos, A., Tzoumas, K., Babu, S. (Eds.), Proceedings of the Third Workshop on Data analytics in the Cloud, DanaC 2014, June 22, 2014, Snowbird, Utah, USA, In conjunction with ACM SIGMOD/PODS Conference, ACM. pp. 13:1–13:4. URL: https://doi.org/10.1145/2627770.2627776, doi:10.1145/2627770.2627776.

[26] The Apache Software Foundation, a. Apache Avro. URL: https://avro.apache.org/.

[27] The Apache Software Foundation, b. Apache Hadoop. URL: https://hadoop.apache.org/.

[28] The Apache Software Foundation, c. Apache Kafka. URL: https://kafka.apache.org/.

[29] The Apache Software Foundation, d. Apache Spark. URL: https://spark.apache.org/.

[30] TopQuadrant, Inc., 2003. TopQuadrant Technology Briefing - Semantic Technology. Technical Report. URL: https://lists.oasis-open.org/archives/regrep-semantic/200402/pdf00000.pdf.

[31] Veeckman, C., Jedlicka, K., De Paepe, D., Kozhukh, D., Kafka, S., Colpaert, P., Cerba, O., 2017. Geodata interoperability and harmonization in transport: a case study of open transport net. Open Geospatial Data, Software and Standards 2, 3. URL: https://doi.org/10.1186/s40965-017-0015-6, doi:10.1186/s40965-017-0015-6.

[32] W3C, . Sparql query language for rdf. URL: https://www.w3.org/TR/rdf-sparql-query/.

[33] Yuhanna, N., 2019. Big Data Fabric 2.0 Drives Data Democratization. Technical Report. Forrester Research, Inc.