Towards Executable Knowledge Graph Translation

Dongzhuoran Zhou[1,2,*], Baifan Zhou[2], Zhuoxun Zheng[1,4], Zhipeng Tan[1],
Egor V. Kostylev[2] and Evgeny Kharlamov[1,2]

[1]*Bosch Center for Artificial Intelligence, Germany*
[2]*Department of Informatics, Univeristy of Oslo, Norway*
[3]*Department of Computer Science, Oslo Metropolitan University, Norway*

Abstract
Data analytics is vital in manufacturing for extracting insights from production data and optimising production processes. Semantic technologies including knowledge graphs (KG) proved to be beneficial for addressing challenges of transparency and explainability of analytics by offering standardised means to describe manufacturing domains, data, analytical tasks and solutions. In this work we discuss executable KGs for industrial analytics; they can be "translated" (i.e. transformed) to executable data pipelines in a reusable and modularised fashion. In particular, we discuss how to capture analytical solutions in the form of data pipelines as KGs, and how to translate such KGs to executable data pipelines. The poster presents our framework, implementation, and preliminary industrial evaluation.

Keywords
knowledge graph, welding monitoring, machine learning, industrial application analytics,

1. Introduction

Data analytics is vital in manufacturing for extracting insights from production data and optimising production processes. Semantic technologies including knowledge graphs (KG) proved to be beneficial for challenges of transparency and explainability [1] of analytics [2] by offering standardised means [3, 4] to describe manufacturing domains [5, 6], data analytical tasks and solutions, as well as robot positioning controlling [7, 8].

In particular, KGs allow to represent executable data analytical pipelines with standardized and formal description to represent the steps in the data pipeline [9]. This opens the door for KG based verification, reasoning, and optimisation, data construction [10], data mining [11, 12, 13] in manufacturing [14]. Consider an industrial scenario, where a multi-disciplinary team including engineers, data scientists, managers work together on quality prediction with ML in car industry [15]. Our project experience reals that the experts with distinct background spent excessive time on discussion but found out they misunderstood the problem and the ML solutions. After that, we tried to use KG as a medium for communication, see an example ML pipeline KG in Fig. 1. It takes *TimeSeries* and *SingleFeatures* as the input data, and does *LRRegression* to predict the *Q-Value*. The users can simply change the input data, output data, and method of the pipeline, by changing the named individuals, e.g., the users can delete *TimeSeries* if they do not have the sensor curves (time series) in their data, because the sensor curves are costly to collect. The users can also change the ML method from *LRRegression* to *MLP* (multilayer perceptron).

Hangzhou'22: The 21st International Semantic Web Conference, October 23–27, 2022, Hangzhou, China
*Corresponding author.
dongzhuoran.zhou@de.bosch.com (D. Zhou)

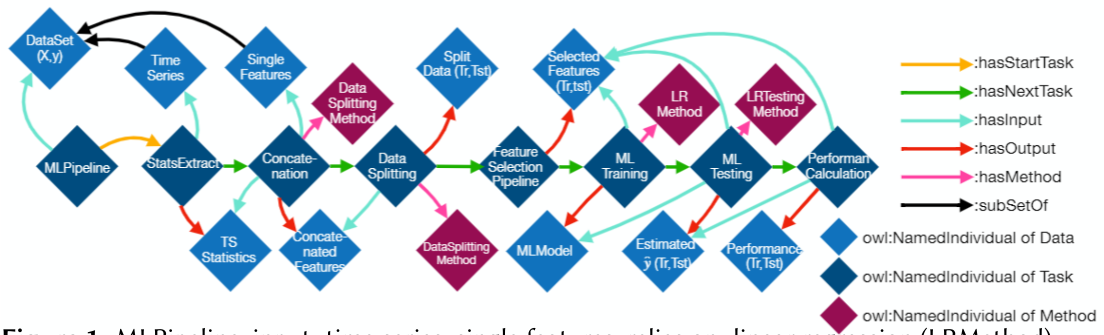

Figure 1: MLPipeline, input: time series, single features; relies on: linear regression (LRMethod).

In this work we study how KGs can facilitate analytics by addressing two issues: how to encode data pipelines as KGs – we refer to such KGs as executable KGs – and how to automatically compute executable pipelines from executable KGs. Here the latter should ensure the correctness of the analytical methods; the desired order of tasks within executable pipelines; the correct number of input and output and corresponding designed dimensions. Moreover, such computation should support typical executable pipelines for visual and statistical analytics [16, 17]. Finally, such executable pipelines should be suitable for large-scale deployment [18].

In this poster paper we exemplify our solution within the domain of quality monitoring in automatic manufacturing. In particular, in this poster we present the following. We present our executable Knowledge Graph framework, its verification, translation and execution methods; we discuss implementation of our solution, and present its preliminary evaluation with Bosch manufacturing data. The experimental results show that our proposed approach is promising in coverage and scalability aspects. This poster accompanies our In-Use track paper accepted at ISWC'22 [19] and gives significant extension on technical details of KG translation.

2. Our Approach

Executable Knowledge Graph Framework. We propose framework for executable KG (ExeKG) that represents ML solutions [20] for solving ML questions. Framework supports ExeKG to be translated to executable scripts and modularised in reusable and modularised fashion.

We first define data, methods and tasks in this framework. *Data* $\mathcal{D}$ is a set of facts, statistics, or items of information, it can be in forms such as numerals, diagrams or strings organised in different structures, typically relational tables or RDF database, etc. A *Method* $\mathcal{F}$ is a function in form of language-dependent script (such as in C++ or Python). A method takes some data which fulfils certain *Constraints* $\mathcal{C}_{\mathcal{F}}$ as input and can output specific data. Formally, $\mathcal{D}_{out} = \mathcal{F}(\mathcal{D}_{in})$, if $\mathcal{C}_{\mathcal{F}}(\mathcal{D}_{in}) = True$. A *Task* $\mathcal{T}$ is the process of invoking a method by feeding it with some data that meets certain *Constraints*, and by doing so to obtain some other data. Formally, $\mathcal{T}\langle\mathcal{D}_{in},\ \mathcal{F}\rangle = \mathcal{F}(\mathcal{D}_{in}) = \mathcal{D}_{out}$, if $\mathcal{C}_{\mathcal{F}}(\mathcal{D}_{in}) = True$. We call each single *Task* as Atomic task.

Some tasks have methods which are unified, while other more complex tasks can not solved by invoking a single integrated method while can be unfolded into a sequence of tasks where each task is a part of the complex one. We refer to the complex tasks as pipelines $\mathcal{T}_p$. Formally, a pipeline $\mathcal{T}_p$ with input data $\mathcal{D}_{in}$ to get $\mathcal{D}_{out}$, expressed as $\mathcal{T}_p\langle\mathcal{D}_{in},\ \mathcal{F}\rangle = \mathcal{D}_{out}$ can be unfolded in the sequence $\{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_n\}$, where:

$$\mathcal{T}_1\langle\mathcal{D}_{in_1},\ \mathcal{F}_1\rangle = \mathcal{D}_{out_1}, \mathcal{D}_{in_1} \subset \mathcal{D}_{in},\ \mathcal{C}_{\mathcal{F}_1}(\mathcal{D}_{in_1}) = True;\ ... \tag{1}$$

$$\mathcal{T}_n\langle\mathcal{D}_{in_n},\ \mathcal{F}_n\rangle = \mathcal{D}_{out_n}, \mathcal{D}_{in_n} \subset \dot{\bigcup}_{i\in\{1,...n-1\}} \mathcal{D}_{out_i} \cup \mathcal{D}_{in},\ \mathcal{C}_{\mathcal{F}_n}(\mathcal{D}_{in_n}) = True$$

$$\longrightarrow \mathcal{D}_{out} \in \dot{\bigcup}_{i\in\{1,...,n\}} \mathcal{D}_{out_i},\ \mathcal{C}_{\mathcal{F}} = \bigcap_{i\in\{1,...,n\}} \mathcal{C}_{\mathcal{F}_i}(\mathcal{D}_{in_i}). \tag{2}$$

Table 2
Categories of the executable KGs (ExeKG). The structure refers to whether there is only one *sequential* data pipeline in the executable KG or there are multiple *parallel* data pipelines. *Atomic task* refers to tasks that cannot be decomposed to smaller tasks, while *Pipeline task*s are comprised from pipelines of multiple atomic tasks. *Multiple input/output* specifies whether the tasks in the data pipeline can take multiple input/ouput or not.

Complexity Type	Structure	Atomic Task/Pipeline Task	Multiple input/output
Linear ExeKG	Sequential	Only Atomic	No
Multilinear ExeKG	Sequential	Only Atomic	Yes
Integrated ExeKG	Sequential	Integrated Pipeline	Yes or No
Parallel ExeKG	Parallel	Only Atomic	Yes
Parallel Integrated ExeKG	Parallel	Integrated Pipeline	Yes

Verification. We use Boolean query (4) and the axioms in KG to offer the correct *Constraints* of translated executable data analytics. A DataEntity is the class for a concrete dataset or a feature. An example can be *Every Task has at least one output data, which is a DataEntity (3-4)*:

$$\forall x.task(x) \rightarrow \exists y(hasOutput(x, y) \wedge DataEntity(y)) \tag{3}$$

$$QUERY : Q(x) \leftarrow Task(x) \wedge \neg\exists y.(hasOutput(x, y) \wedge DataEntity(y)) \tag{4}$$

Translation and Execution. The translation can be discussed with two structures of executable KGs:

1) *Sequential:* here each executable KG is in the form of a *Pipeline*, which consists of a series of *Tasks* of sequential structures connected with *hasNextTask*. Thus, the translation of an executable KG invokes the Python function scripts with the inputs/outputs and parameters given by *DataEntity* and datatype properties of KGs, according to the order defined by *hasNextTask*.

Table 1: Task complexity, categories and coverage.

KG Type	Structure	Avg. #Atomic Tasks	Coverage
VisualKG	Linear	5 to 10	100%
	Multilinear	5 to 10	85%
StatsKG	Linear	1 to 5	100%
	Multilinear	5 to 10	95%
	Parallel	10 to 20	90%
	Integrated	10 to 20	80%
MLKG	Integrated	More than 20	80%
	Parallel Integrated	More than 20	80%

2) *Parallel*: In the case of merging two parallel structures, the translator will search preceding dependency with *hasNextTask*, until no preceding *Task* is found.

Implementation. We implemented a system for executable KG translation [21] with three functional modules: 1) *Databases* including the *relational database* and its APIs and RDF database, 2) Analytics module and 3) KG processing module. Fig. 2a shows the structure of the system. The *Databases* are responsible for storing the welding data and executable KGs. Its APIs handles the loading, formatting, filtering, padding and merging of different data subsets. The *Analytics module* stores and provides interface for all analytical methods of Visual KG, StatsKG and ML KG in Python scripts. The *KG processing module* performs verification of the executable KGs, translates the executable KGs to executable pipelines and executes these pipelines by connecting the analytical methods stored in the Analytics module.

3. Evaluation and Conclusion

Transparency and Coverage Evaluation. *Theoretical discussion*: We formulate visual, statistical and ML analytics in the form of Eq. 1-3. These three forms provide a way to describe analytics tasks in a general and straightforward way, which eases the understanding of the tasks, since the users only need to understand the description once and then they can

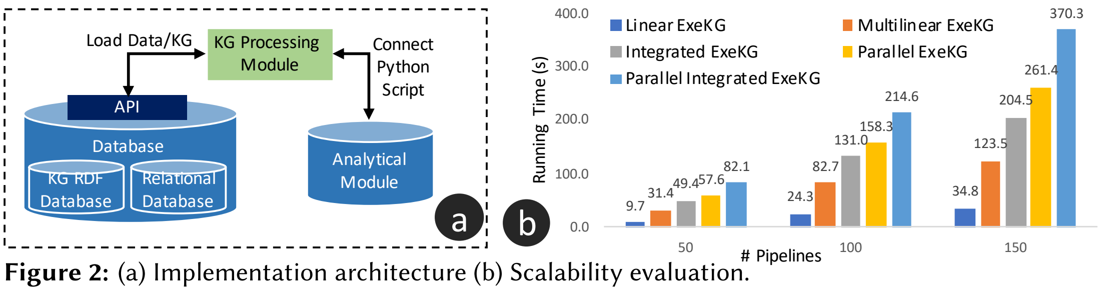

Figure 2: (a) Implementation architecture (b) Scalability evaluation.

understand similar data analytical pipelines described in this way. Thus, our approach is a step towards more transparent way that cover cases of data analytical pipelines described with Eq. 1-3. *Empirical evaluation*: We organised extensive workshops with the ML and non-ML experts. After discussion, we categorised most tasks of visual, statistical and ML analytics encountered in our project in groups (see Table 1), and give the coverage percentage according to our empirical cases. Observe, for all of the cases the coverage is above 80%, for some of the cases even above 90%. Besides, the users also gave their subjective evaluation on transparency with questionnaires, where they answered questions such as "*I found the Executable KGs make data analytics easier to understand* " and gave scores ranging 1-5 (ranging for disagree to agree). The average score was 4.28 ± 0.47 (mean ± standardeviation) which shows good transparency.

Scalability Evaluation. We evaluate the scalability of our approach by the running time of translation and execution of executable KGs with different complexity type (Fig. 2b).

Data Description. To have controllable scope, we tested these executable KGs on a sample welding production dataset collected from a German factory. The dataset is in relational tables form after integration, containing 4585 welding operation records, 2 welding programs, performed by 1 welding machine and deals with 2 types of car bodies.

Results and Discussion. Fig. 2b demonstrates that our system scales well since it takes limited time to translate executable KGs to scripts and execute scripts. On most right hand side, we see that the translation and execution of most complex executable KGs, namely parallel integrated executable KG, only takes 6 minutes for 150 KGs, on the given data, which shows good scalability.

Conclusion and Outlook. In this poster we present our ongoing research of representing data analytical pipelines in KGs and transformation (also called "translation") of such KGs in executable analytical pipelines. We discussed framework, verification, translation and execution with our scope of welding monitoring with a Bosch case and evaluated our approach with real industrial data and users from Bosch case, which shows promising results. In the future, we plan to generalise our approach to more cases and to host the system regularly on the Bosch environment and constantly collect more user feed-backs. We will also study more technical details such as expressivity and limitations in theory and practice, and compare with other similar work such as Yahoo! Pipes, DAGs in Spark, Tez, the PROV-O ontology.

Acknowledgments

The work was partially supported by H2020 projects Dome 4.0 (Grant Agreement No. 953163), OntoCommons (Grant Agreement No. 958371), DataCloud (Grant Agreement No. 101016835) and the SIRIUS Centre, Norwegian Research Council project number 237898.

References

[1] B. Mahesh, Machine learning algorithms-a review, IJSR 9 (2020) 381–386.

[2] Z. Zheng, et al., Executable knowledge graph for transparent machine learning in welding monitoring at bosch, in: CIKM, 2022.

[3] B. Motik, et al., OWL 2 web ontology language profiles, 2012. URL: https://www.w3.org/TR/owl2-profiles/.

[4] A. Salatino, et al., The computer science ontology: a large-scale taxonomy of research areas, in: International Semantic Web Conference, 2018, pp. 187–205.

[5] J. Davis, et al., Smart manufacturing, manufacturing intelligence and demand-dynamic performance, Computers & Chemical Engineering 47 (2012) 145–156.

[6] B. Zhou, et al., Exploiting the values of data: A holistic semantification approach at Bosch, in: ESWC (Demos/Industry), Springer, 2022.

[7] C. Naab, et al., Application of the unscented kalman filter in position estimation a case study on a robot for precise positioning, RAS 147 (2022) 103904.

[8] O. Celik, et al., Specializing versatile skill libraries using local mixture of experts, in: CRL, PMLR, 2022, pp. 1423–1433.

[9] M. Yahya, et al., Towards generalized welding ontology in line with ISO and knowledge graph construction, in: ESWC (Posters & Demos), 2022.

[10] D. Zhou, et al., Towards Ontology Reshaping for KG Generation With User-In-The-Loop: Applied to Bosch Welding, in: IJCKG, 2021.

[11] B. Zhou, et al., Scaling Usability of ML Analytics With Knowledge Graphs: Exemplified with A Bosch Welding Case, in: IJCKG, 2021.

[12] D. Zhou, et al., Enhancing knowledge graph generation with ontology reshaping–bosch case, ESWC (Demos/Industry), Springer (2022).

[13] K. K. Breitman, et al., Ontology in computer science, Semantic Web: Concepts, Technologies and Applications (2007) 17–34.

[14] D. Zhou, et al., ScheRe: Schema Reshaping for Enhancing Knowledge Graph Construction, in: CIKM, 2022.

[15] B. Zhou, Y. Svetashova, A. Gusmao, A. Soylu, G. Cheng, R. Mikut, A. Waaler, E. Kharlamov, SemML: Facilitating development of ML models for condition monitoring with semantics, Journal of Web Semantics 71 (2021) 100664.

[16] B. Zhou, et al., Knowledge graph-based semantic system for visual analytics in automatic manufacturing, in: ISWC, 2022.

[17] Z. Zheng, et al., Towards a visualisation ontology for data analysis in industrial applications, in: ESWC, 2022.

[18] D. Zhou, et al., Ontology reshaping for knowledge graph construction: Applied on bosch welding case, in: ISWC, 2022.

[19] Z. Zheng, et al., Executable Knowledge Graphs for Machine Learning: A Bosch Case of Welding Monitoring, in: ISWC, Springer, 2022.

[20] P. M. LaCasse, et al., A Survey of Feature Set Reduction Approaches for Predictive Analytics Models in the Connected Manufacturing Enterprise, Applied Sciences 9 (2019).

[21] Z. Zheng, et al., ExeKG: Executable Knowledge Graph System for User-friendly Data Analytics, in: CIKM, 2022.