LLM-Supported Manufacturing Mapping Generation

Wilma Johanna Schmidt[1]
Corporate Research, Robert Bosch GmbH, Renningen, Germany
AG Corporate Semantic Web, Freie Universität Berlin, Germany

Irlan Grangel-González
Corporate Research, Robert Bosch GmbH, Renningen, Germany

Adrian Paschke
Data Analytic Center (DANA), Fraunhofer Institute FOKUS, Berlin, Germany
AG Corporate Semantic Web, Freie Universität Berlin, Germany

Evgeny Kharlamov
Corporate Research, Robert Bosch GmbH, Renningen, Germany

Abstract

In large manufacturing companies, such as Bosch, that operate thousands of production lines with each comprising up to dozens of production machines and other equipment, even simple inventory questions such as of location and quantities of a particular equipment type require non-trivial solutions. Addressing these questions requires to integrate multiple heterogeneous data sets which is time consuming and error prone and demands domain as well as knowledge experts. Knowledge graphs (KGs) are practical for consolidating inventory data by bringing it into the same format and linking inventory items. However, the KG creation and maintenance itself pose challenges as mappings are needed to connect data sets and ontologies. In this work, we address these challenges by exploring LLM-supported and context-enhanced generation of both *YARRRML* and *RML* mappings. Facing large ontologies in the manufacturing domain and token limitations in LLM prompts, we further evaluate ontology reduction methods in our approach. We evaluate our approach both quantitatively against reference mappings created manually by experts and, for *YARRRML*, also qualitatively with expert feedback. This work extends the exploration of the challenges with LLM-supported and context-enhanced mapping generation *YARRRML* [18] by comprehensive analyses on *RML* mappings and an ontology reduction evaluation. We further publish the source code of this work. Our work provides a valuable support when creating manufacturing mappings and supports data and schema updates.

2012 ACM Subject Classification Computing methodologies → Semantic networks

Keywords and phrases Mapping Generation, Knowledge Graph Construction, Ontology Reduction, RML, YARRRML, LLM, Manufacturing

Digital Object Identifier 10.4230/TGDK.3.3.5

Category Use Case

Supplementary Material
Software (Source code, example data set): `https://github.com/boschresearch/myamr_tgdk` [19]

Funding *Evgeny Kharlamov*: Partially supported by the EU Projects Graph Massivizer (GA 101093202), enRichMyData (GA 101070284), and SMARTY (GA 101140087).

Acknowledgements The authors would like to thank Ishan Dindorkar for his support in setting up the technical environment.

Received 2025-06-02 **Accepted** 2025-11-17 **Published** 2025-12-10

Special Issue Use-Case Articles

[1] corresponding author

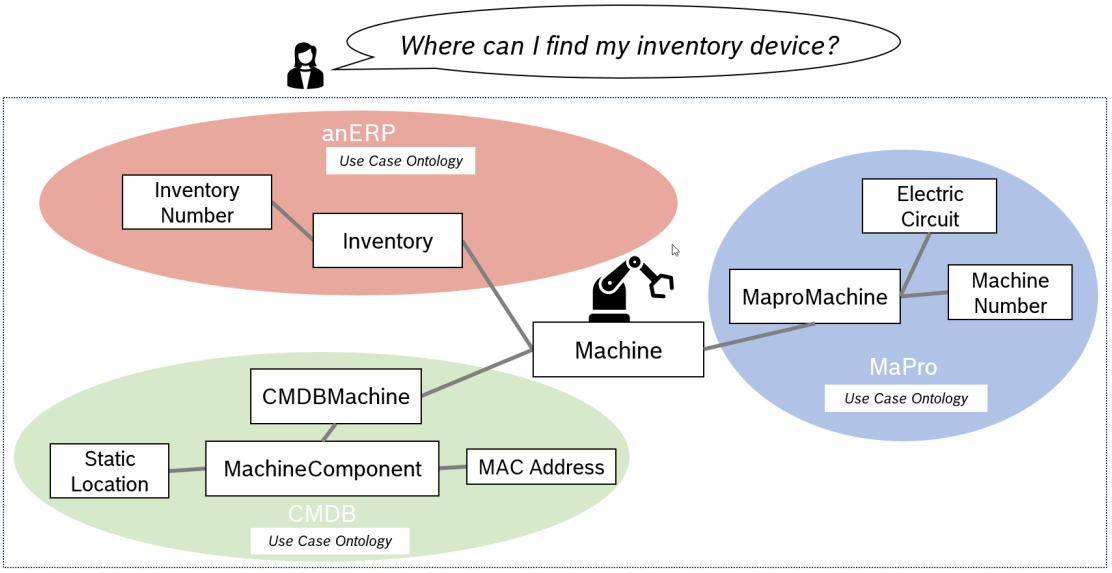

Figure 1 Motivational question and illustration of distributed data. Data belonging to the same machine is stored in different, use case specific sources for an enterprise resource planning, configuration management, and manufacturing project management.

1 Introduction

In large manufacturing companies, such as Bosch, data related to a specific machine inventory item is stored in distributed databases, which are built and curated to serve the needs for one or multiple functions within the company, see Figure 1. Cross-functional questions, e.g., linking physical location and financial information, often lead to *treasure hunting* with laborious human efforts, experts' alignments, and delays when looking for the answer. While efforts increase with growing data volumes, the pressure on fast and sound decisions in the manufacturing domain rises.

The semantic approach has proven to be successful in bringing heterogeneous data to a common format and interlinking datasets, e.g., tables from relational databases, with the help of ontologies and Knowledge Graphs (KGs). Indeed, it has already been adopted in various industrial sectors such as energy [12], manufacturing [17], e-commerce [1] and others. In particular, Bosch has been relying on ontologies and KGs in many applications ranging from analyses of discrete manufacturing [29] to autonomous driving [22] and combinatorial optimization [5]. Moreover, Bosch has been scaling the use of semantics in various large projects such as the European data infrastructure project *Gaia-X*[2] and its derivative *Catena-X*.

In these applications the key task is to link a domain ontology to the original data with the help of declarative mappings and then execute the mappings in order to create a domain KG. The creation and maintenance of manufacturing mappings is non-trivial. Especially, domain-specific vocabulary, updates in data and data structure as well as complex facts and relations across different data silos pose challenges. As of now, it is common practice to create them manually. For this, a domain expert and a knowledge expert are required and they align during the mapping generation. Mappings must be conform to the ontology of the KG. The domain expert understands which real object or concept is actually referenced by the label in the data set, whereas the knowledge expert retrieves related ontologies, understands the ontology syntax, and creates syntactically correct mappings.

[2] https://gaia-x.eu/news-press/the-role-of-ontologies-in-gaia-x/

YARRRML has been introduced as a human-readable representation of RDF mappings and provides an alternative to the standard mapping languages *Relational Database to RDF Mapping Language (R2RML)*[3] and *RDF Mapping Language (RML)*[4]. Van Assche *et al.* [23] assess in a systematic literature review, *eight* mapping languages excluding *YARRRML*, noting that "it was excluded because it is published as a demo", yet acknowledging its widespread use in practice. We can second the industrial use of *YARRRML*, e.g., at several plants at Bosch, due to its vendor independence as well as further enhancements (see e.g. [10], [24]). Van Assche *et al.* [23] assess *RML* which is an extension of the W3C standard language *R2RML* from databases only to further structured sources such as *.csv* or *.json*. High quality mappings are of core importance for a functional manufacturing KG, yet their creation remains, with both *RML* or human-friendly *YARRRML*, challenging. We explore mapping generation on each of the two languages in our work.

LLMs in a manufacturing context often work with unstructured data such as text (e.g., error descriptions) or images (e.g., optical inspection of parts). Research on neuro-symbolic KG construction in manufacturing ([20]) shows that most such approaches are conducted on unstructured data sources. Exploration of neuro-symbolic AI approaches for creating mappings for KG construction in manufacturing is still missing. As LLMs with their well-known strength in generating text, and also code, from unstructured text rapidly evolve, it is promising to explore their capabilities of dealing with structured data, e.g., mapping tasks [11]. However, varying accuracy, token size limitations and missing domain-focus of LLMs require advanced generation workflows and configurations to realize the applicability of this approach. Retrieval-Augmented Generation (RAG) approaches and neuro-symbolic architectures such as semantically context-enhanced prompts can increase accuracy and domain-focus. Additionally, ontology reduction methods show means of reducing the prompt size, while potentially improving accuracy. We explore this in our work. Our guiding question is:

> *Can LLM-supported and context-enhanced mapping generation support human expert mapping creation for manufacturing KGs?*

This work focuses on addressing this question and the contributions of this paper are summarized as follows:

1. We provide the first exploratory approach on LLM-supported and context-enhanced mapping generation for KGs as support for knowledge experts in a manufacturing company.
2. We propose a novel combination of context-based ontology reduction and a RAG-based approach for automatic mapping generation to create manufacturing KGs.
3. We follow a pragmatic evaluation approach for mapping quality on manufacturing KGs and apply our method on several configurations against gold-standard mapping references which are created by domain and technical experts. By this, we obtain comprehensive insights on influencing factors for high quality manufacturing mapping creation.

The further parts of this paper are structured as follows: We introduce the use case and existing challenges in Section 2. In Section 3, we propose our approach for manufacturing mapping generation and explain the experimental implementations. We present our results on LLM-supported and context-enhanced mapping generation and lessons learned in Section 4 and briefly discuss related work in Section 5. We conclude with future work in Section 6.

[3] `https://www.w3.org/TR/r2rml/`, `https://www.w3.org/2001/sw/wiki/RDB2RDF`
[4] `https://rml.io/specs/rml/`

2 Use Case

In this section, we outline the use case and challenges in manufacturing inventory identification. Currently in Bosch, process experts and data analysts have no common shared and accessible data base that provides information about machines executing the same process. Data items for a single machine are spread across different silos. If the relevant information is available and enriched with various other influencing factors, process experts can roll out solutions easier and faster. For instance, to be able to answer the following question: "How many winding machines are there in a certain plant"? data from different sources needs to be integrated. To that end, a KG has been built to semantically integrate and harmonize data to cope with such questions. Next, we list the main competency questions (*CQs*) for the KG creation and maintenance. The questions are a prioritized excerpt from a list of questions from practice. The questions were originally collected by domain experts for the overall ontology- and graph-based approach and the selected questions address all three manufacturing ontologies from our scenario. For validation of their currentness, the questions have additionally been reviewed by a subject expert. The *CQs* are:

- **CQ 1:** How many machines of a specific type are located in a given plant?
- **CQ 2:** Where is the machine inventory located that is allocated on my cost center?
- **CQ 3:** Is the insecure machine control component still used in other stations within the plant?
- **CQ 4:** What are all financial information to a line with only the machine identifier given?
- **CQ 5:** What is the unique identifier number to a CMDB asset?

Over time, changes are done in the current data set structure and further data sets will need to be added, so that answers to these questions are up-to-date: the KG supports these tasks pragmatically. For this, mapping generation and updates are needed. With growing data volumes and ontology sizes as well as the need for more efficiency, (semi-)automated mapping generation approaches are needed. However, multiple challenges occur in manufacturing mapping generation in practice if mappings are generated automatically, e.g., with the support of LLMs:

- **Challenge 1:** How to feed a manufacturing ontology into a prompt?
- **Challenge 2:** Can context-based ontology reduction methods improve mapping generation for manufacturing KGs?
- **Challenge 3:** Can prompt context enhancements improve the quality of the generated mapping?
- **Challenge 4:** Which sample size of a data set is needed for a generated mapping of high quality?

Next, we describe the data sources required to answer the main competency questions.

2.1 Data sources

In this part, we describe the data sources of *MaPro*, *CMDB*, and *anERP* in detail, which are identified as relevant for our example.

MaPro. The Manufacturing Project Management (*MaPro*) is a data source of the special machinery that covers project management details, e.g., timeline and capacity planning. It contains information for mechanical construction, e.g., machines of a production line and the link to the plans.

CMDB. The Configuration Management Database (*CMDB*) is a database for documenting and linking hardware and software assets, i.e., configuration items, e.g., users, relationships between assets or IP addresses used. In the context of this work it provides, e.g., the IP and MAC address from the machines in the manufacturing environment.

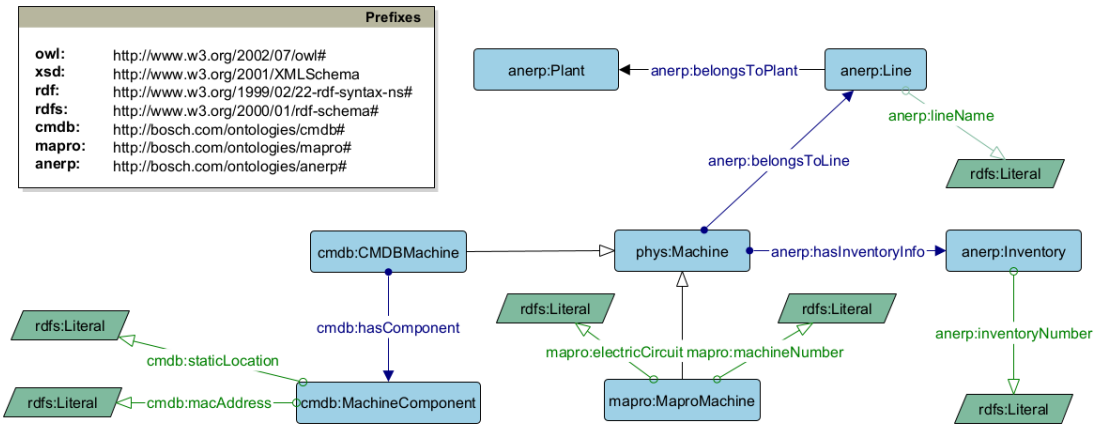

Figure 2 Main classes and properties of the MaPro, CMDB, and anERP ontologies. Illustration of our use case relevant classes, properties and property types, and the object properties. Prefixes are shown as supplements.

anERP. The *anERP* is an analytics platform for enterprise supply chain data. For this particular project, the anERP source enables the collection of financial data of the assets, i.e., machines and machine components. Furthermore, it contains inventory data of the assets.

2.2 Ontologies

In this section, we describe the structure of the ontologies used to build the KG. While the ontologies are core for our mapping generation (cf. Figure 2), they are proprietary and we may not share them publicly. However, we will provide quantifiable details on these ontologies in Section 3. The three ontologies employed in this work model the domains of datasets *MaPro*, *CMDB*, and *anERP*, respectively. The ontologies are based on the Core Information Model for Manufacturing (CIMM) ontologies [8] and comparable in their structure as they describe a similar domain. The CIMM ontologies provide a framework to reuse most important concepts from the manufacturing domain, e.g., machines, components, plants, lines, among others[5]. While each of our ontologies targets the corresponding dataset-relevant perspective from the manufacturing domain, they are rather simple w.r.t. hierarchy, number of classes etc. as their main intention is to describe the data from the data sources. They all share the formalization in the sublanguage *Description Logics (DL)* of the *Web Ontology Language (OWL)*[6]. Our ontologies comprise entries of type `owl:Class`, `owl:ObjectProperty`, and `owl:DatatypeProperty`. While classes mainly contain a label information, both property types may contain next to label, type, and comments also domain and range restrictions.

3 Approach for LLM-Supported and Context-based Mapping Generation

In this section, we introduce *MYAM+R*, a **M**anufacturing **YA**RRRML **M**apping generation approach, which also support *RML* mappings. It comprises four input modules, an ontology reduction module, the core mapping module which generates a mapping, an evaluation module

[5] `https://github.com/eclipse-esmf/esmf-manufacturing-information-model`
[6] `https://www.w3.org/TR/owl-features/`

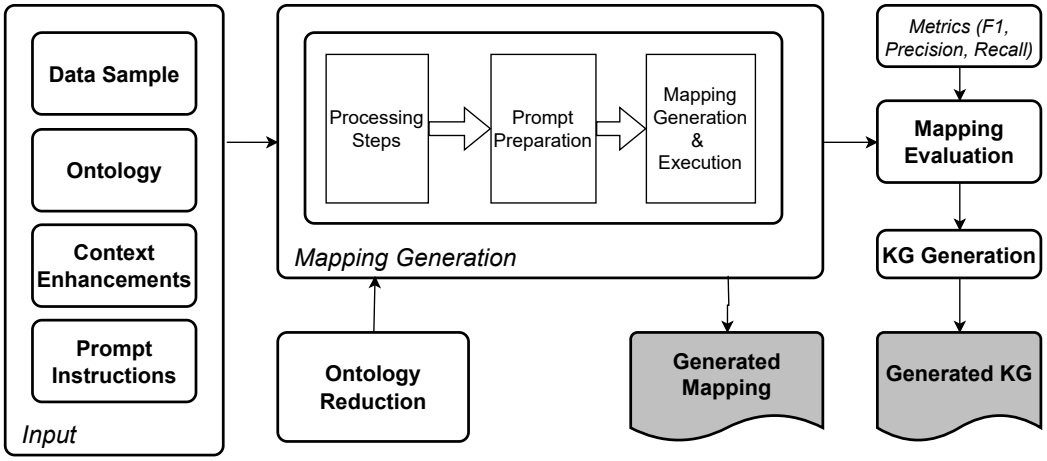

Figure 3 Manufacturing Mapping Generation Architecture (*MYAM+R*) for *YARRRML* and *RML* mappings. Illustration of the *MYAM+R* modules for input (data sample, ontology, context enhancements, and prompt instructions), ontology reduction, mapping generation, evaluation, and KG generation.

and a KG generation module. In Figure 3 we show an overview of *MYAM+R*. In the following, we explain all modules and process steps. We conclude this section with implementation details of *MYAM+R*.

3.1 *MYAM+R* Modules

First, we introduce the input modules which provide (1) a data sample, (2) an ontology, (3) prompt context enhancements, and (4) prompt instructions.

Data Sample. This module provides data from a relational database, described in Section 2.1, in *csv* format. The data size configuration, i.e., the portion of the data used for mapping generation, is specified with a parameter in *MYAM+R*. *MYAM+R* retrieves the full data set and then crops it accordingly to the configured setting. An example snippet is given in Figure 7.

Ontology. This module provides an ontology file relevant for the provided data set in *turtle* format. To meet prompt size restrictions, *MYAM+R* contains an ontology reduction module which we introduce later in this section. An example ontology snippet is given in Figure 7.

Enhancements. This module provides a prompt context enhancement as a *txt* file. The enhancements can be of different type, e.g., generic mapping templates or few shot examples. We show generic mapping templates for both *YARRRML* and *RML*, independent of data input in the prompt, yet teaching the syntax, in the following.

We further show generic one-shot examples which cover a correct mapping fragment for a given data input for each mapping language in Figures 5 and 6. In Section 3.2 we provide details on our enhancement implementation. The enhancement configuration is specified with a parameter in *MYAM+R*. The mapping generation module retrieves the context enhancement from this enhancement module based on the parameter value.

Prompt Instructions. This module provides instructions for the final mapping generation prompt.

```
prefixes:
  mach: <http://example.com/MachineOntology#>

mappings:
  machine:
    sources:
      - [examples/data.csv]
    s: mach:$(MachineID)
    po:
      - [a, mach:Machine]
      - [mach:machineId, $(MachineID)]
      - [mach:machineName, $(MachineName)]
      - p: mach:hasComponent
        o:
        - mapping: component

  component:
    sources:
      - [examples/data/machine_component.csv]
    s: mach:$(ComponentID)
    po:
      - [a, mach:Component]
      - [mach:componentId, $(ComponentID)]
      - [mach:componentName, $(ComponentName)]
```

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix mach: <http://example.org/MachineOntology#> .

<#MachineMapping>
    a rr:TriplesMap ;
    rml:logicalSource [
        rml:source "examples/data.csv" ;
        rml:referenceFormulation ql:CSV     ] ;
    rr:subjectMap [
        rr:template
        ↪  "http://example.org/MachineOntology/
        ↪  {MachineID}" ;
        rr:class ont:Machine     ] ;
    rr:predicateObjectMap [
        rr:predicate ont:machineId ;
        rr:objectMap [ rml:reference "MachineID" ] ] ;
    rr:predicateObjectMap [
        rr:predicate ont:machineName ;
        rr:objectMap [ rml:reference "MachineName" ] ]
        ↪  ;
    rr:predicateObjectMap [
        rr:predicate ont:hasComponent ;
        rr:objectMap [
            rr:parentTriplesMap <#ComponentMapping>  ]
            ↪  ] .
<#ComponentMapping>
    a rr:TriplesMap ;
    rml:logicalSource [
        rml:source "examples/data.csv" ;
        rml:referenceFormulation ql:CSV     ] ;
    rr:subjectMap [
        rr:template
        ↪  "http://example.org/MachineOntology/
        ↪  {ComponentID}" ;
        rr:class ont:Component     ] ;
    rr:predicateObjectMap [
        rr:predicate ont:componentId ;
        rr:objectMap [ rml:reference "ComponentID" ]  ]
        ↪  ;
    rr:predicateObjectMap [
        rr:predicate ont:componentName ;
        rr:objectMap [ rml:reference "ComponentName" ]
        ↪  ] .
```

Figure 4 Mapping Templates in *YARRRML* (left) and *RML* (right).

Ontology Reduction. This module takes data and ontology as an input and returns a reduced ontology. This module is needed because the provided manufacturing ontology contains elements which are not relevant for the mapping of the specific input data. This module supports in our implementation *three* reduction approaches, one generic and two content-based ones.

Mapping Generation. This module provides the core functionalities of *MYAM+R* and interacts with all other modules. *MYAM+R* supports different experimental settings and can be configured with parameters for convenient execution. Based on the parameter values, *MYAM+R* preprocesses the data, reduces the ontology, collects a prompt enhancement, and prompt instructions. The mapping generation module creates a prompt template with the collected ingredients as a prompt with input variables and sends this to an LLM. Finally, *MYAM+R* processes and validates the response and sends the generated mapping to the evaluation module. We call this step *mapping generation* as it is building upon the existing, reduced ontology in our previous step. The mapping is generated on the basis of this ontology. So, an alignment and consistency with this fixed ontology are ensured. We validate the generated mapping in an evaluation part (see next step) against a consistent, manually created gold standard mapping.

```
[
    {
        "input_ontology": "@prefix owl: <http://www.w3.org/2002/07/owl#> .
           @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
           @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
           @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
           @prefix ont: <http://example.com/PlantOntology#> .

           ont: rdf:type owl:Ontology ;
           rdfs:label \"plant-ontology\" ;
           .

           ont:Plant
           rdf:type owl:Class ;\
           rdfs:label \"ONT Plant\"@en ;
           rdfs:comment \"Describes the Plant entity in the ONT data base.\"@en .

           ont:plantName
           rdf:type owl:DatatypeProperty ;
           rdfs:label \"plant name\" ;
           rdfs:range xsd:string ;
           rdfs:domain ont:Plant .",
        "input_data": "PlantID,PlantName
        1,Plant_A
        2,Plant_B",
        "result_mappings": "...(see YARRML and RML result mappings figures)..."
    }
]
```

Figure 5 One Shot Example Input Data and Ontology.

Evaluation. This module receives a generated mapping, a reference mapping, and an ontology as input and returns evaluation scores for the generated mapping. The specific evaluation is described in Section 4.

KG Generation. This module generates a KG based on the generated mapping and a data set as input.

3.2 *MYAM+R* Implementation

MYAM+R is implemented in Python 3.13. For the mapping generation execution and for the LLM-based ontology reduction, we choose *gpt-4-turbo (version:1106-Preview)*. Aiming at a solution close to practice, we evaluate on a pre-trained LLM which could be available in large companies. A typical user in such an enterprise could get API access while avoiding steps of deployment and a potential training or fine-tuning of a base model (which would include dependencies on involvement of data specialists). At the same time, the user's task can become easier as he or she benefits from high quality output as most LLMs with higher payment models come with higher performance. Next, we describe the configurations that we explore in our experiments to address the challenges described in Section 2.

Data Sample. We run our experiments on data samples from *MaPro*, *CMDB*, and *anERP*. The metrics of these sets are given in Table 1. We set different configurations for the input: 1) only column headers, 2) column headers and first 20 data value rows, and 3) column headers and first 200 data value rows.

```
"results_mappings":
    "prefixes:
    @prefix ont:
    ↪  <http://example.com/
    ↪  PlantOntology#> .

    mappings:
      plant:
        sources:
          - [examples/
          ↪  data.csv]
    s: ont:$(PlantID)
    po:
      - [a, ont:Plant]
      - [ont:plantName,
      ↪  $(PlantName)]. "
```

```
"result_mappings": "@prefix rr: <http://www.w3.org/ns/r2rml#> .
    @prefix rml: <http://semweb.mmlab.be/ns/rml#> .
    @prefix ql: <http://semweb.mmlab.be/ns/ql#> .
    @prefix ont: <http://example.com/PlantOntology#> .

    <#PlantMapping>\n    a rr:TriplesMap ;
        rml:logicalSource [
            rml:source \"examples/data.csv\" ;
            rml:referenceFormulation ql:CSV   ] ;
        rr:subjectMap [
           rr:template \"http://example.com/PlantOntology#{PlantID}\" ;
           rr:class ont:Plant          ] ;
        rr:predicateObjectMap [
            rr:predicate ont:plantName ;
            rr:objectMap [
                rml:reference \"PlantName\"   ]   ]   ."
```

Figure 6 One Shot Example Result Mappings in *YARRRML* (left) and *RML* (right).

```
PlantID,PlantName,LineID,LineName
1,Plant_A,101,Line_X1
1,Plant_A,102,Line_X2
2,Plant_B,103,Line_Y1
2,Plant_B,104,Line_Y2
3,Plant_C,105,Line_Z1
3,Plant_C,106,Line_Z2
4,Plant_D,107,Line_W1
4,Plant_D,108,Line_W2
5,Plant_E,109,Line_V1
5,Plant_E,110,Line_V2
```

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix ont: <http://example.com/PlantOntology#> .

ont:PlantOntology a owl:Ontology ;
  owl:versionInfo "1.0" .

ont:Plant a owl:Class ;
  rdfs:label "Plant" ;
  rdfs:comment "A plant is a facility where production takes place." .
ont:Line a owl:Class ;
  rdfs:label "Line" ;
  rdfs:comment "A line is a production line within a plant." .
ont:hasLine a owl:ObjectProperty ;
  rdfs:label "has line" ;
  rdfs:comment "A plant has one or more lines." ;
  rdfs:domain ont:Plant ;
  rdfs:range ont:Line .
ont:plantID a owl:DatatypeProperty ;
  rdfs:label "plant ID" ;
  rdfs:comment "A unique identifier for a plant." ;
  rdfs:domain ont:Plant ;
  rdfs:range xsd:integer .
```

Figure 7 Example data source (left) and corresponding ontology snippet (right).

Ontology. The metrics of the corresponding ontologies to the sources are listed in Table 1. The files in our approach have been selected by a knowledge expert.

Context Enhancement. We explore zero shot, with and without a generic *YARRRML* mapping template, and few shot examples. The zero shot example with template contains a *YARRRML* mapping independent of the data source. The few shot examples are data-source- and ontology-specific and are manually created by domain and knowledge experts. For each data source we use one file of examples in *JSON* format. We implement the few shots with a RAG approach; *MYAM+R* retrieves the example file, vectorizes the examples as strings, embeds and stores them into a vector store. We select with the preprocessed, i.e., potentially cropped input data, and reduced ontology only the one most similar example and further process it in *MYAM+R*.

Prompt Instructions. Following common guidelines, we apply concise prompt instructions for the LLM to generate a *YARRRML* mapping. The instructions include a role *("You are a Senior Manufacturing Knowledge Expert...")*, an input description, and specific instructions depending on the shot variant. E.g., a zero shot prompt with a template states that the template is only

Table 1 Metrics of data sets and ontologies. *Key: DP – Datatype Properties, OP – Object Properties.*

Data Source	Data Set		Ontology			
	Columns	Rows	Classes	DP	OP	Namespaces
MaPro	17	2,215	4	13	4	5
CMDB	16	18,082	5	15	2	5
anERP	17	15,451	3	7	2	5

relevant for syntax, whereas a few shot prompt suggests to also learn from the content of the given mapping for the target one. The instructions are data- and ontology-independent. The prompt instructions are included in the published repository[7] to support reproducibility.

Ontology Reduction Methods. Ontology summarization [27], modularization [13], and matching [15] are important research topics[8], yet for the sake of pragmatic mapping generation we do not dive deeper into the current state of research. Instead, we address this in future work, and next, describe *three* reduction approaches which we evaluate in our experiments.

- **Naive:** It simplifies ontologies to specific elements associated with the input data. In our case, only the classes with their properties `rdf:type`, `rdfs:label`, and `rdfs:comment` as well as datatype and object properties remain in the ontology. In addition, we bind all prefixes and namespaces to the reduced ontology. Finally, all potential four consecutive empty space characters are removed from the reduced ontology.
- **Similarity-based:** It reduces the ontology to those elements (and a neighborhood) which have a similarity above a specified threshold with the column names from the data source. For this, ontology and data set embeddings are needed. We embed both ontology and data set with the transformers model *distilbert-base-uncased* [16]. *Distilbert* is a light, performant version compared to the *BERT* model and a common approach. We apply cosine similarity and bind namespaces and prefixes after identifying the most similar elements.
- **LLM-based:** It receives input data with column headers and first *200* data value rows as well as the ontology as prompt enhancement and returns a reduced ontology. We additionally extract the ontology from surrounding natural language response, if needed, see 4.7, and evaluate the syntax, see Section 4.

Next, we explain the implementation of the mapping generation module.

Mapping Generation. The embedding of the vectorized few shot examples is implemented with the model *sentence-transformers/paraphrase-multilingual-mpnet-base-v2* from an instance of *HuggingFaceEmbeddings*. For the vector store we use the *FAISS* library for easy semantic similarity search. To select the best shot example, we use an instance of the *SemanticSimilarityExampleSelector*. Prior to the evaluation, the mapping module extracts the mapping from any surrounding natural language, similar to the step in *LLM-based ontology reduction*. *MYAM+R*, hence, only evaluates the generated mapping. Next, *MYAM+R* checks the LLM response against *YARRRML* syntax by storing the file. If this returns an error, the syntax is considered invalid. If the file is successfully stored, it is considered valid. If valid, *MYAM+R* evaluates the mapping quality with the *YARRRML* evaluation module which compares the generated triples with the reference ones in relaxed mode. In addition, we conducted a qualitative *MYAM+R* expert evaluation. The results and evaluation steps are presented in the next section.

[7] `https://github.com/boschresearch/myamr_tgdk`
[8] See e.g., Workshop Ontology Matching (`https://om.ontologymatching.org/2024/`) as part of ISWC 2024: `https://iswc2024.semanticweb.org/`

```
prefixes:
    ont: http://example.com/PlantOntology#

mappings:
    plant:
        sources:
            - [plants.csv-csv]
        s: ont:Plant/$(PlantID)
        po:
            - [a, ont:Plant]
            - [ont:plantID, $(PlantID)]
            - [ont:plantName, $(PlantName)]
```

```
prefixes:
  ont: http://example.com/PlantOntology#

mappings:
  plant:
    sources:
      - [examples/data_sources/csv/generic/data/
        ↪ plant_line_machine.csv]
    s: ont:plant-$(PlantID)
    po:
      - [a, ont:Plant]
      - [ont:plantID, $(PlantID)]
      - [ont:plantName, $(PlantName)]
```

Figure 8 Comparison of snippets of a generated mapping (left) and gold standard mapping (right) of example class *plant* in *YARRRML*. A difference in *IRI* can be seen.

KG Generation. The original data set and generated mapping are input to generate an RDF KG in *turtle* format. For this, each triple is populated with the corresponding data via IDs in the triple with column names, e.g., a mapping triple subject with ID *$(PlantName)* in the *IRI* is populated with *Plant_A*, *Plant_B*, *Plant_C*, *Plant_D*, and *Plant_E* from the data in Figure 7.

4 Evaluation

In this section, we show our experimental results and evaluate the outcome, both quantitatively and qualitatively. With multiple setting configurations, we address the challenges described in Section 2 to achieve mapping generations with high quality and reduced prompt size. We evaluate the generated mappings against gold standards with *F1*-score, *precision* and *recall*. Lastly, we collect feedback on our approach and outcome. We first describe the gold standard mapping.

4.1 Gold Standard Mappings

Our *YARRRML* reference mappings are gold standard mappings and created manually by domain and knowledge experts. They are applied in practice for mapping generation. For *RML* evaluation, we manually map these *YARRRML* mappings to *RML* while preserving the mapping logic. The reference mappings represent the standard way in which an RDF KG is generated, if data source, ontology, and mappings are available. The reference mappings contain only the triples which are relevant for mapping the respective input data.

4.2 Metrics

We are evaluating experiments with different configurational settings for mapping generation. We assess the quality of each generated mapping, specifically the triples, w.r.t. the reference mapping. The reference triples are correctly generated by experts. All triples generated with *MYAM+R* are called *positives*, all other triples *negatives*. A correct triple generation is labeled with *true*, an incorrect one with *false*. The first interesting value are the *true positives*, which is the number of correct triple generations. However, errors may occur. A *false positive* is a generated triple which is not in the reference set. A *false negative* triple is not generated by the model, although it is in the reference set. The remaining *true negatives* are irrelevant and not analyzed. To evaluate each experiment, we employ F1-score (*F1*), precision (P) and recall (R). P represents the number of true positives in relation to overall positives by the model. R represents the number of true positives in relation to all positives. Finally, *F1* is the harmonic mean of P and R and formally defined as: $F1 = \frac{2}{\frac{1}{P}+\frac{1}{R}}$.

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix ql: <http://semweb.mmlab.be/ns/ql#> .
@prefix rdfs:
↪ <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ont: <http://example.com/PlantOntology#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<#PlantMapping> a rr:TriplesMap ;
    rml:logicalSource [
        rml:source "data.csv" ;
        rml:referenceFormulation ql:CSV ] ;

    rr:subjectMap [
        rr:template "http://example.com/
        ↪ PlantOntology#Plant_{PlantID}" ;
        rr:class ont:Plant ] ;

    rr:predicateObjectMap [
        rr:predicate ont:plantID ;
        rr:objectMap [
            rml:reference "PlantID" ;
            rr:datatype xsd:integer ] ] ;

    rr:predicateObjectMap [
        rr:predicate ont:plantName ;
        rr:objectMap [
            rml:reference "PlantName" ;
            rr:datatype xsd:string ] ] .
```

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix ont: <http://example.com/PlantOntology#> .

<#PlantMapping>
    a rr:TriplesMap ;
    rml:logicalSource [
        rml:source
        ↪ "examples/data_sources/csv/data.csv" ;
        rml:referenceFormulation ql:CSV
    ] ;
    rr:subjectMap [
        rr:template
        ↪ "http://example.com/PlantOntology/
        ↪ {PlantID}" ;
        rr:class ont:Plant
    ] ;
    rr:predicateObjectMap [
        rr:predicate ont:plantId ;
        rr:objectMap [ rml:reference "PlantID" ]
    ] ;
    rr:predicateObjectMap [
        rr:predicate ont:plantName ;
        rr:objectMap [ rml:reference "PlantName" ]
    ] .
```

Figure 9 Comparison of snippets of a generated mapping (left) and gold standard mapping (right) of example class *plant* in *RML*. Differences in prefix declaration, source and subject *IRI* can be seen.

4.3 Mapping Generation Evaluation

Next, we examine how far the challenges described in Section 2 are addressed and, whether specific *MYAM+R* configurations can give an efficient support for experts during mapping generation. As prework for our evaluation criteria, we discuss known quality criteria for RML mappings [11] and *YARRRML* mappings [14]. Moreau *et al.* [14] update in their work dimensions and metrics for RML mapping evaluations while leaning on proposed dimensions and metrics from Zaveri *et al.* [26] on quality assessments for linked data. Hofer *et al.* [11] evaluate LLM-generated RML-mappings and the resulting KGs. The focus of their evaluations is on "fitness for use" [11], e.g., with relaxed triple matches, which compare predicate and object of a triple, but only the contained *ID* and type of a subject instead of the exact *IRI*. Similar to Hofer *et al.*, the majority of Moreau *et al.*'s dimensions and metrics is not directly applicable to our requirements as (1) LLM-supported mapping evaluation differs to the assessment of manually created mappings, (2) gold standard mapping evaluation differs to measuring mapping quality in general, and (3) our objective of expert support differs from fully automated generations. We focus on criteria to support experts best ("fitness of use" [11]) instead of a comprehensive mapping quality assessment. First, we validate the syntax of any ontology or mapping generated by an LLM. Additionally, we conduct a comparison of *(i)* the subontology against a reference ontology (manually created based on input ontology and reference mapping) and *(ii)* of generated with reference triples. Hofer *et al.* realize the generated triple evaluation on several layers with exact and relaxed scores on triples, subject, predicate, objects, properties and more. We focus on the relaxed match of generated triples as our sources provide no constraints on *IRI* construction and the data has complex assumptions on building *IRI*s. Instead of evaluating a variety of mapping aspects, we focus on those which give insights into the choice of configurational settings for *MYAM+R* to support experts on mapping generation. We evaluate our approach on the following criteria: $\boldsymbol{C_1}$**:** Valid reduced ontology syntax *(only necessary for LLM-based reduction)*, $\boldsymbol{C_2}$**:** Match of prefixes, classes, datatype properties, and object properties of generated subontology against reference ontology, $\boldsymbol{C_3}$**:** Valid generated

Table 2 Mapping Evaluation on All Configurations and Data Sets for *YARRRML*. Evaluation (*precision (P)*, *recall (R)*, *F1-Score (F1)*) of relaxed match of generated mapping triples. *Key: Config. – Configuration, D – Data Size, O – Ontology Reduction, E – Context Enhancement, dnf – did not finish.*

Configuration (C)				MaPro			CMDB			anERP			Mean		
ID	DS	OR	CE	P	R	F1	P	R	F1	P	R	F1	P	R	F1
1	columns	naive	zero	0.80	0.53	0.64	0.35	0.47	0.40	0.75	**0.90**	0.82	0.63	0.63	0.62
2		naive	template	0.86	0.80	0.83	0.38	0.40	0.39	0.67	0.80	0.73	0.63	0.67	0.65
3		naive	few	dnf	dnf	dnf	0.53	0.67	0.59	0.75	0.30	0.43	0.64	0.48	0.55
4		similar	zero	0.58	0.73	0.65	0.50	0.67	0.57	0.75	**0.90**	0.82	0.61	0.77	0.68
5		similar	template	0.69	0.60	0.64	0.47	0.53	0.50	0.80	0.80	0.80	0.65	0.64	0.65
6		similar	few	0.80	0.80	0.80	dnf	dnf	dnf	0.64	0.70	0.67	0.72	0.75	0.73
7		LLM	zero	0.50	0.60	0.55	0.64	0.46	0.54	0.64	0.70	0.67	0.59	0.59	0.59
8		LLM	template	0.62	0.53	0.57	0.67	0.53	0.60	0.63	0.50	0.56	0.64	0.52	0.57
9		LLM	few	0.67	0.53	0.59	0.67	0.53	0.60	0.63	0.50	0.56	0.65	0.52	0.58
10	first20	naive	zero	0.75	0.80	0.77	0.40	0.53	0.46	0.67	0.80	0.73	0.61	0.71	0.65
11		naive	template	0.79	0.73	0.76	0.65	**0.73**	0.69	0.82	**0.90**	**0.86**	0.75	**0.79**	**0.77**
12		naive	few	0.63	0.80	0.71	**1.00**	0.27	0.42	0.60	0.30	0.40	0.74	0.46	0.57
13		similar	zero	0.74	**0.93**	0.82	0.40	0.53	0.46	0.67	0.80	0.73	0.60	0.76	0.67
14		similar	template	0.75	0.80	0.77	0.61	**0.73**	0.67	0.73	0.80	0.76	0.70	0.78	0.73
15		similar	few	0.63	0.80	0.71	0.55	**0.73**	0.63	0.67	0.80	0.73	0.62	0.78	0.69
16		LLM	zero	0.53	0.53	0.53	0.67	0.53	0.60	0.64	0.70	0.67	0.61	0.59	0.60
17		LLM	template	0.73	0.53	0.62	0.57	0.53	0.55	0.88	0.70	0.78	0.72	0.59	0.65
18		LLM	few	0.67	0.53	0.59	0.70	0.47	0.56	0.80	0.80	0.80	0.72	0.60	0.66
19	first200	naive	zero	0.58	0.73	0.65	0.58	**0.73**	0.65	0.58	0.70	0.64	0.58	0.72	0.64
20		naive	template	0.73	0.73	0.73	0.73	**0.73**	**0.73**	0.80	0.80	0.80	0.76	0.76	0.76
21		naive	few	**1.00**	0.20	0.33	**1.00**	0.27	0.42	0.75	0.30	0.43	0.92	0.26	0.40
22		similar	zero	0.55	0.73	0.63	0.50	0.67	0.57	dnf	dnf	dnf	0.53	0.70	0.60
23		similar	template	0.69	0.73	0.71	0.44	0.53	0.48	0.80	0.80	0.80	0.64	0.69	0.67
24		similar	few	0.81	0.87	**0.84**	**1.00**	0.27	0.42	0.75	0.30	0.43	0.85	0.48	0.61
25		LLM	zero	0.64	0.60	0.62	0.64	0.47	0.54	0.80	0.80	0.80	0.69	0.62	0.66
26		LLM	template	0.64	0.47	0.54	0.70	0.47	0.56	0.60	0.60	0.60	0.65	0.51	0.57
27		LLM	few	**1.00**	0.20	0.33	**1.00**	0.27	0.42	**0.86**	0.60	0.71	**0.95**	0.36	0.52

mapping syntax, and $\boldsymbol{C_4}$**:** Relaxed match of generated mapping triples against gold standard (on *F1*, *P*, and *R*). We give examples of a generated mapping class, *plant*, and the respective reference mapping class for *YARRRML* in Figure 8 and for *RML* in Figure 9.

4.3.1 YARRRML Results

We refer to the experimental results in 2 with *Y-* and ID number, e.g., *Y-1* for the first experiment (data size columns-only, naive ontology reduction, zero shot context) conducted. Our mapping evaluation consists of two phases. First, we assess the syntax correctness of the generated mappings as described in Subsection 3.2. This is an important step as LLMs, specifically *GPT-models* [6] show difficulties on this task. The mapping syntax is valid in all *81* runs except for *three* experiments, *Y-3*, *Y-6*, and *Y-22*, which are marked as *dnf* in Table 2. Each invalid run originates from a duplicate key in the generated mapping. Hence, C_3 is partially fulfilled. Next, we evaluate the generated mapping triples relaxed against the reference and show our results in Table 2. Configuration *Y-27* achieves highest precision across all data sets (*P=0.95*) and also on each

Table 3 Mapping Evaluation on All Configurations and Data Sets for *RML*. Evaluation (*precision (P)*, *recall (R)*, *F1-Score (F1)*) of relaxed match of generated mapping triples. *Key: DS – Data Size, OR – Ontology Reduction, CE – Context Enhancement.*

Configuration (C)				MaPro			CMDB			anERP			Mean		
ID	DS	OR	CE	P	R	F1	P	R	F1	P	R	F1	P	R	F1
1	first20	naive	template	0.65	**0.68**	0.67	0.76	0.62	0.68	**0.75**	**0.75**	**0.75**	0.72	**0.68**	**0.70**
2		naive	few	0.72	**0.68**	0.70	0.74	0.67	0.67	0.73	0.67	0.70	0.73	0.67	**0.70**
3		similar	template	0.48	0.53	0.50	0.65	0.62	0.63	**0.75**	**0.75**	**0.75**	0.63	0.63	0.63
4		similar	few	0.61	0.58	0.59	0.67	0.67	0.67	0.73	0.67	0.70	0.67	0.64	0.65
5		LLM	template	0.60	0.47	0.53	0.71	0.48	0.57	0.73	0.67	0.70	0.68	0.54	0.60
6		LLM	few	0.60	0.47	0.53	0.86	0.57	0.69	0.70	0.58	0.64	0.72	0.54	0.62
7	first200	naive	template	0.71	0.63	0.67	0.61	0.52	0.56	0.50	0.50	0.50	0.61	0.55	0.58
8		naive	few	**0.81**	**0.68**	**0.74**	0.78	0.67	0.72	0.64	0.58	0.61	**0.74**	0.64	0.69
9		similar	template	0.50	0.53	0.51	0.57	0.57	0.57	**0.75**	**0.75**	**0.75**	0.61	0.62	0.61
10		similar	few	0.53	0.53	0.53	0.83	**0.71**	**0.77**	0.70	0.58	0.64	0.69	0.61	0.64
11		LLM	template	0.73	0.42	0.53	0.83	0.48	0.61	0.57	0.33	0.42	0.71	0.41	0.52
12		LLM	few	0.55	0.32	0.40	**0.88**	0.33	0.48	**0.75**	0.50	0.60	0.72	0.38	0.49

individually. Overall, *Y-11* shows best *recall* (*R=0.78*, with even *R=0.93* on data set *MaPro*), and overall best *F1 (0.77)*. We observe that all best *precision* values come from few shot configurations. All configurations show most difficulties on data set *CMDB*.

4.3.2 RML Results

We refer to the experimental results in 3 with *R-* and ID number, e.g., *R-1* for the first experiment (data size first20, naive ontology reduction, template context enhancement) conducted. Following our evaluation structure explained in the previous section, we first assess the syntax correctness of the generated mappings as described in Subsection 3.2. The mapping syntax is valid in all *36* runs, so, C_3 is fulfilled. Next, we evaluate the generated mapping triples relaxed against the reference and show our results in Table 3. Note that we build upon our learnings from [18] on *YARRRML* results with poor performance from datasize columns-only and context enhancement zero shot. Hence, we execute *RML* experiments on configurations with datasize first20 or first200, with context enhancement mapping template or few shot, and with all ontology reduction methods. In our experiments, we encountered in *four* out of *36* experiments rate size limits with our *gpt* model. Hence, *R-12* are conducted on only 150 instead of 200 rows for the *MaPro* dataset and *R-8*, *R-10* as well as *R-12* are conducted on only 100 instead of 200 rows. Configuration *R-8* achieves highest precision across all data sets (*P=0.74*) caused mainly by the precision value from the *MaPro* dataset. The highest precision on an individual dataset is achieved by *R-12* which is the LLM-based ontology reduction with few shot examples on the first200, but due to rate limit issues only on first 100 data rows from CMDB. Overall, *R-1* shows best *recall* (*R=0.68*, and, together with *R-2* overall best *F1 (0.70)*. With a slight deviation from the *YARRRML* results, we observe that best *precision* values may come from few shot or template configurations. We see varying result performance on an approach across datasets. E.g., *R-12* shows low performance on *MaPro* (e.g., P=0.55), yet achieves highest precision values compared to the other approaches on *CMDB* (0.88) and on *anERP* (0.75, shared with *C9*).

Table 4 Ontology Reduction Method Evaluation on All Configurations and Data Sets for *RML*. Evaluation (*F1-Score (F1)*) of classes, datatype properties, and object properties against reference ontology. *Key: OR – Ontology Reduction.*

OR		MaPro					CMDB					anERP					Mean Method		
		c-F1	d-P	d-R	d-F1	o-F1	c-F1	d-P	d-R	d-F1	o-F1	c-F1	d-P	d-R	d-F1	o-F1	c-F1	d-F1	o-F1
naive	*1*	1.0	0.83	1.0	0.91	1.0	1.0	0.87	1.0	0.93	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.95	1.0
	2	1.0	0.83	1.0	0.91	1.0	1.0	0.87	1.0	0.93	1.0	1.0	1.0	1.0	1.0	1.0			
	3	1.0	0.83	1.0	0.91	1.0	1.0	0.87	1.0	0.93	1.0	1.0	1.0	1.0	1.0	1.0			
	4	1.0	0.83	1.0	0.91	1.0	1.0	0.87	1.0	0.93	1.0	1.0	1.0	1.0	1.0	1.0			
similar	*1*	1.0	0.83	1.0	0.91	1.0	1.0	0.87	1.0	0.93	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.95	1.0
	2	1.0	0.83	1.0	0.91	1.0	1.0	0.87	1.0	0.93	1.0	1.0	1.0	1.0	1.0	1.0			
	3	1.0	0.83	1.0	0.91	1.0	1.0	0.87	1.0	0.93	1.0	1.0	1.0	1.0	1.0	1.0			
	4	1.0	0.83	1.0	0.91	1.0	1.0	0.87	1.0	0.93	1.0	1.0	1.0	1.0	1.0	1.0			
LLM	*1*	0.89	0.89	.80	0.84	0.67	1.0	0.75	0.46	0.57	0.0	0.8	1.0	0.57	0.73	0.67	0.88	0.73	0.43
	2	0.75	0.86	0.60	0.71	0.40	1.0	0.88	0.53	0.67	0.67	0.8	1.0	0.71	0.83	0.67			
	3	0.75	0.86	0.60	0.71	0.40	1.0	0.75	0.46	0.57	0.0	1.0	1.0	0.86	0.92	0.0			
	4	0.89	0.89	0.80	0.84	0.67	0.67	0.83	0.38	0.53	0.0	1.0	1.0	0.86	0.92	1.0			

4.4 Evaluation of Ontology Reduction Methods

In the following, we analyse the *three* ontology reduction methods introduced in Subsection 3.2. With the results, we address *Challenge 2* raised in Section 2. We validate the ontology syntax manually. For *YARRRML*, in all *27* experiments, the LLM-based reduced ontology is correct (C_1 fulfilled). For *RML*, in all *12* experiments, the LLM-based reduced ontology is correct (C_1 fulfilled). We show the *F1-score* of all three reduction methods on all datasets from the *RML* runs in Table 4. Note that the mapping language has no influence on the ontology reduction as it is not part of the input for this step. You may not map the IDs to the previous *RML* IDs as we merely stress the fact of four runs per ontology method at this point. All reduction methods perform well when retrieving relevant classes. Only in exceptions, specifically in more than half of the LLM-based runs, not all relevant classes were retrieved, hence, a lower *recall* led to a slight drop in *F1*-score. We analyse the datatype properties in more detail as datasets and approaches show different results. *Naive* and *similar* reduction perform perfect on *anERP*. This is also the reason for best performance of these two approaches overall (*F1* of classes: 1.0, *F1* of datatype properties: 0.95, *F1* of object properties: 1.0). The LLM-based ontology reduction methods shows difficulties in comparison to these two methods, especially with an overall *F1*-score of only 0.43 for object properties. Hence, C_2 is partially fulfilled.

4.5 Evaluation of Knowledge Graph Generation

Next, we present the KG generation results for *YARRRML* and *RML*. For *YARRRML*, we show the best performing configurations (with $F1 \geq 0.60$) on mapping generation in Table 5. The generated KGs contain tens of thousands of triples with different subjects which underlines the necessity of mappings and supported mapping generation. Configuration *Y-2* generates the most triples across all data sets and especially outperforms for data set *anERP*. The most triples on *MaPro* are generated by *Y-19*, for *CMDB* by *Y-1*, despite their average performances on mapping generation on these data sets. The reason for this lies in a high match on the subject *IRI*s which are not represented in the relaxed triple score. In *nine* cases, a configuration did not generate a KG because either the mapping was not generated in the previous step or due to an error such

Table 5 Generated KG Triples from *YARRRML* on Configurations with *mean F1* $\geq$ *0.60* in mapping generation. Total triples (TT), unique subjects (S), unique predicates (P), classes (C), datatype properties (D), object properties (O). Missing configurations are marked as (–) (not generated).

ID	MaPro						CMDB						anERP						Mean
	TT	S	P	C	O	D	TT	S	P	C	O	D	TT	S	P	C	O	D	TT
1	11,399	2,708	11	**5**	1	10	**66,544**	9,704	16	**4**	1	15	79,963	11,869	10	**3**	1	9	52,635
2	13,099	2,708	15	**5**	**5**	10	34,883	1,994	15	3	2	13	**162,232**	**11,951**	**17**	**3**	**3**	**14**	**70,071**
4	15,918	2,708	15	**5**	**5**	10	63,865	9,704	**17**	**4**	1	**16**	68,338	11,869	10	**3**	1	9	49,374
5	12,718	2,304	12	4	3	9	59,323	**9,824**	16	**4**	**3**	13	69,017	**11,951**	10	**3**	**3**	7	47,019
6	11,401	2,766	11	**5**	1	10	–	–	–	–	–	–	–	–	–	–	–	–	11,401
10	10,884	2,708	12	**5**	1	**11**	–	–	–	–	–	–	80,822	**11,951**	11	**3**	**3**	8	45,853
11	19,594	4,030	14	**5**	**5**	9	63,912	9,704	16	**4**	**3**	13	59,543	**11,951**	9	**3**	2	7	47,683
13	15,914	2,708	15	**5**	4	**11**	–	–	–	–	–	–	59,572	**11,951**	9	**3**	2	7	37,743
14	15,284	2,304	**16**	**5**	**5**	**11**	65,754	9,704	**17**	**4**	**3**	14	80,822	**11,951**	10	**3**	**3**	7	53,953
15	15,862	2,748	15	**5**	**5**	10	63,860	9,704	**17**	**4**	1	**16**	69,017	**11,951**	10	**3**	1	9	49,580
16	–	–	–	–	–	–	39,178	9,704	9	**4**	1	8	56,766	11,869	9	**3**	**3**	6	47,972
17	6,705	1,763	10	4	3	7	53,253	9,704	13	**4**	**3**	10	47,773	**11,951**	8	**3**	**3**	5	35,910
18	8,289	2,304	9	4	1	8	35,659	7,867	8	3	2	6	79,668	11,869	9	**3**	**3**	6	41,205
19	**21,895**	**4,518**	15	**5**	**5**	10	62,169	9,704	16	**4**	1	15	60,608	**11,951**	10	**3**	**3**	7	48,224
20	–	–	–	–	–	–	60,388	9,704	14	**4**	**3**	11	69,017	**11,951**	10	**3**	**3**	7	64,703
22	15,284	2,304	**16**	**5**	**5**	**11**	–	–	–	–	–	–	–	–	–	–	–	–	15,284
23	15,261	2,285	16	**5**	**5**	**11**	62,532	9,704	**17**	**4**	**3**	14	69,017	**11,951**	10	**3**	**3**	7	48,937
24	11,452	2,748	12	**5**	1	**11**	27,651	7,710	4	1	1	3	–	–	–	–	–	–	19,552
25	10,508	2,392	10	**5**	4	6	18,033	4,644	9	3	1	8	37,329	**11,951**	8	**3**	2	6	21,957

as non-retrievable *ID*s from the mapping. For *RML*, we show all configurations from the *RML* mapping generation in Table 6. The generated KGs contain tens of thousands of triples with different subjects which underlines the necessity of mappings and supported mapping generation. Configuration *R-4* (which corresponds to *R-4 in Table 6*) generates the most triples across all data sets. Continuing our running example, we finally show a snippet of generated RDF triples in Figure 10 with our generated RML mapping on the example data and ontology from the previous figures.

4.6 Qualitative Evaluation

As a final step, for the *YARRRML* mapping, we collect feedback from *two* domain and *two* knowledge experts on *MYAM+R* regarding effectiveness, overall satisfaction, and open feedback. The feedback on the first *two* categories is shown in Figure 11. We see that the experts on average agree to strongly agree that the generated mappings and the generated KG support their work. They also agree that they are satisfied with the approach overall and are neutral to agreeing to recommend *MYAM+R* to their colleagues. One main reason for a hesitance in recommendation to colleagues is a missing user-friendly frontend, which is, in addition to improving the provided context and *MYAM+R* documentation noted as areas of improvement by the experts. We will address this in future work. As strengths of *MYAM+R* the experts see handling large, complex ontologies and source tables with many columns. They further highlight the time savings and data quality improvement with *MYAM+R*.

Table 6 Generated KG Triples from all *RML* mapping generations. Total triples (TT) and unique subjects (S).

ID	MaPro		CMDB		anERP		Mean	
	TT	**S**	**TT**	**S**	**TT**	**S**	**TT**	**S**
1	**17,739**	**4,396**	38,472	6,480	**68,338**	11,869	41,516	7,582
2	11,450	2,410	60,474	**9,704**	68,043	11,869	46,656	7,994
3	13,201	2,474	62,098	**9,704**	**68,338**	11,869	47,879	**8,016**
4	11,608	2,410	**63,998**	**9,704**	68,043	11,869	**47,883**	7,994
5	9,084	2,020	48,858	**9,704**	56,713	11,869	38,218	7,864
6	9,079	2,018	43,691	8,026	68,043	11,869	40,271	7,304
7	11,893	2,472	35,221	3,613	63,518	11,869	45,204	8,015
8	8,135	2,410	54,228	**9,704**	68,043	11,869	41,293	7,435
9	11,339	2,107	35,221	3,613	**68,338**	11,869	38,299	5,863
10	8,455	1,973	54,228	**9,704**	67,800	11,869	43,494	7,849
11	9,784	2,381	25,424	6,481	56,173	11,868	30,460	6,910
12	9,064	1,970	35,192	7,711	57,036	**11,950**	33,764	7,210

4.7 Discussion and Lessons Learned

As data preparation is an issue, we chose *csv* to have a harmonization of the data set appearances. Domain-specific vocabulary and abbreviations in data such as *P_No*, which represents the project number of a machine, can be difficult for LLMs. Further, in some ontology elements we encounter unfinished comments and typos, e.g., `rdfs:comment "Describes the Plant entitiy [...]"`, which pose challenges in understanding. To directly access the relational data sources, *ontology-based data access (OBDA)* is a well-known paradigm as it encompasses "a semantic paradigm for providing a convenient and user-friendly access to data repositories" [25] including relational data bases. Our work supports the *OBDA* approach by having the ontology and source data to support users by creating mappings more rapidly and highlighting important points in the data.

In test runs, we observe multiple times a "chattiness" in LLM responses, i.e., a wrapping in natural language around the ontology or mapping, e.g., "Certainly, here is your mapping: ". This happens despite explicit instructions to *(i) not respond in natural language, but only return the mapping* and to *(ii) not include any explanations in your response*. Hence, we implement a processing step to remove any pattern surrounding the desired artefact. While the ontologies listed in our use case are rather small, in general, they are much larger in practice since they reuse domain and upper ontologies. We address this in future work.

When executing the experiments, we learn that with configurations with a high mean *F1*, we are able to generate KGs which can answer, e.g., *How many machines of a specific type are located in a particular plant? (CQ1)*. In Section 2, we list the main competency questions, yet in reality, there are many more. With our best performing mapping configurations and resulting KGs, these questions can be answered. While the *YARRRML* mapping configurations with the best *precision* utilize few shot prompts, we encounter *two YARRRML* experiments with few shots, specifically with columns only and naive reduction, that do not finish in generating a mapping. Further, *one* zero shot configuration on *200* data lines and similarity reduction leads to a non finishing run, which motivates us to conduct more runs in future work for further insights. Best precision values for *RML* mappings originate from template or few shot enhancements (we do not evaluate zero shot for *RML*). All *RML* experiments finish in generating a mapping and also, generating triples.

```
@prefix ns1: <http://example.com/PlantOntology#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ns1:Line_101 a ns1:Line ;
    ns1:hasLine ns1:Plant_1 ;
    ns1:lineId 101 ;
    ns1:lineName "Line_X1" .

ns1:Line_102 a ns1:Line ;
    ns1:hasLine ns1:Plant_1 ;
    ns1:lineId 102 ;
    ns1:lineName "Line_X2" .

ns1:Plant_1 a ns1:Plant ;
    ns1:hasLine ns1:Plant_2 ;
    ns1:plantID 1 ;
    ns1:plantName "Plant_A" .
```

Figure 10 Generic example snippet of generated triples from a generated RDF mapping, generic input data and ontology with *MYAM*.

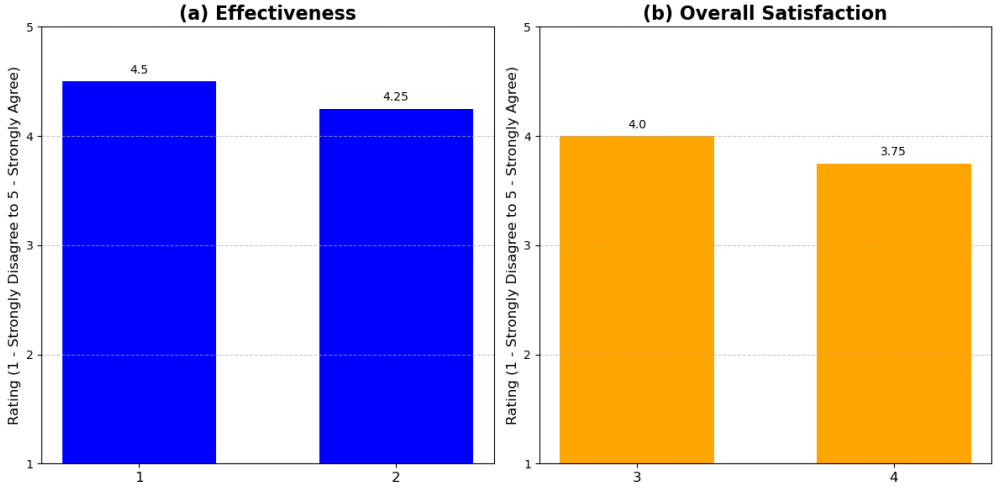

Figure 11 Expert Feedback on *MYAM+R*. Average answer on *(a)* effectiveness of and *(b)* satisfaction with *MYAM+R*. *Key: 1 – The mappings..., 2 – The KGs generated with MYAM+R provide support for experts on manufacturing mapping generation, 3 – Overall, I am satisfied with the MYAM+R approach, 4 – I would recommend MYAM+R to other experts for the task of mapping and KG generation.*

When analyzing the exact triples match, the results drop significantly for both mapping languages. The reason lies in the challenges of *IRI* generations that match the reference mapping. The column *ID* is in the mapping, yet the exact string often does not match the one in the reference mapping. This can be corrected by a knowledge expert with low effort, so that *MYAM+R* overall shows good support for mapping generation tasks. Nevertheless, improving the quality of LLM responses and handling the prompt size are still open challenges.

Clear limitations of our work are, first, the skipped step of accessing the data directly from the relational DB instead of a manually reviewed *.csv* file. The research on this task is active (see e.g., *TEXT2SPARQL*[9]) and it is interesting to incorporate state-of-the-art approaches in the future for

[9] `https://text2sparql.aksw.org/` co-located with Text2KG at ESWC 2025.

this step. Second, an evaluation on different LLMs is missing as well as an in-depth cost analysis which impacts a potential adoption of our approach in practice. Specifically, considering potential future regulatory updates for industry, we will expand our work in next steps with evaluations on other LLMs including open weight models. Further, we evaluate only on our proprietary data set as a public in-use benchmark set for this problem is missing. We publish a generic running example for reproducibility, however, the gap to our real industrial data remains.

Last but not least, one core motivation is the reduction of expert efforts during KG construction. Assessing these efforts in *MYAM+R*, we identify via the expert feedback an effort reduction compared to the process implemented currently in practice in our example plant. Instead of starting mapping generation from scratch, experts can in practice start from a *MYAM+R*-based mapping proposal, review the mapping and update it accordingly. However, we do not believe that experts' guidance or reviews will become completely obsolete in a target state due to the dynamic complexity of industrial data landscapes.

5 Related Work

In this section, we discuss relevant work on LLM-supported mapping generation in manufacturing. First, we review works on automatic mapping generation for KG creation and their applicability to the manufacturing domain, as well as briefly consider schema-supported mapping generation. Second, we discuss the current state of research on LLM-based KG construction. Our approach of using data input, ontology and instructions is comparable to a Graph RAG architecture as the steps of enriching an LLM prompt with schema information to generate an output, in our case a mapping, are comparable to the components of a Graph RAG. However, there is one major distinction: as our goal is KG construction, we must not assume existing supporting graph instance data in our framework.

The research field of automatic KG construction has been popular for a long time with new recent approaches. Zhong et al. [28] state in their corresponding survey (2024) that "[l]arge pre-trained models have been leading a significant impact on multiple KG construction tasks and their related applications". Our focus on *Mapping Generation* corresponds to the step of *Knowledge Acquisition* which mainly covers *Entity Discovery* and *Relation Extraction* for (semi-)structured data sources according to [28] as they also include tables in their analysis (however, without focus on them). The authors state that rule-based extraction methods "are the general solutions for NER" [28], with *Named Entity Recognition (NER)* being the first step of identifying and extracting entities from a data source as part of *Entity Discovery*. In the manufacturing domain however, the disadvantage of this approach lies in the challenge of identifying the correct pairs of (i) source column name and (ii) corresponding entity in the ontology as these pairs may differ drastically on a lexicographical level. Hence, experts conduct this decision manually or, for overall manufacturing KG construction, other "advanced deep learning methods such as BiLSTM-CRF, BiLSTM-ALBERT, etc." [21] are utilized. While experts' time is rare and costly, deep learning methods require training to prepare a model for the task at hand. Both approaches, hence, are not optimal for the requirements of efficiency and scalability in manufacturing. Chen et al. [2] state that for domain KGs "existing KG construction methods heavily rely on human intervention to attain qualified KGs, which severely hinders the practical applicability in real-world scenarios". The authors exploit LLMs for KG construction – focussing on extraction from text – and achieve a result surpassing precision values of existing SOTA KG construction methods.

While multiple KG construction approaches acquire knowledge from unstructured data sources, *Mapping Generation* from structured sources such as relational databases differs, e.g., in the availability of a structure (schema). Utilizing this schema during knowledge extraction can prove helpful, as seen in e.g., Medeiros *et al.* [3] who introduce MIRROR, a system to generate R2RML

mappings. One mapping type produces RDF triples homomorphic to the ones generated by a DM approach [3]. Another results in additional mappings containing implicit knowledge from database schemas [3]. Also, Hazber *et al.* [9] integrate the database schema R2RML mapping generation.

Next, we review first approaches on LLM-supported mapping generations. Hofer *et al.* explore LLM-supported RML mapping generation. The authors see promising results of commercial LLMs, such as *Claude3 Opus (20240229)* and *GPT4 Turbo (01-25-preview)* with zero-shot and on a subset from the Internet Movie Database. Our approach differs from these works in three aspects: (1) we not only explore *RML*, but further *YARRRML* syntax, (2) we investigate necessary ontology reduction, and (3) we evaluate the performance on real manufacturing data from a plant. There exist benchmark tasks [6] for LLM performance evaluation on KG engineering tasks, e.g., parsing and creating KGs serialized in Turtle syntax. Frey *et al.* [7] assess the evolution in Turtle skills of selected commercial LLMs on such tasks. While they show an improvement on the majority of tasks of the LLMs over their predecessors, the authors observe different responses from the models, especially on the output format, and conclude that stricter multi-shot tests are worth evaluating. They further see an open question on the evaluation on RML related tasks. We address both aspects for *RML* and *YARRRML* syntax. Thoroughly assessing the mapping quality is needed from a practical manufacturing perspective, yet research shows gaps in this area. Dimou *et al.* [4] introduce a quality assessment and a refinement workflow for RDF mappings. There is further a "framework to assess the quality of RDF mappings" [14] in *YARRRML* syntax, yet, the set of metrics is not validated against an existing ontology as in our use case. Some approaches on LLM-supported and context-enhanced mapping generations have been proposed and case-study-explored [11]. To the best of our knowledge, none of the approaches have been applied in a real manufacturing setting and none considers the *YARRRML* syntax. The previous works do not take an existing target ontology into consideration, which is a constraint in our setting. As a result, state-of-the-art solutions are insufficient to address our use case. We address these research gaps in our work.

6 Conclusion and Outlook

In this paper, we introduce *MYAM+R*, an approach for LLM-supported and context-enhanced *YARRRML* and *RML* mapping generation, as a support for manufacturing knowledge experts. *MYAM+R* comprises of *eight* modules: *four* for input, as well as one for ontology reduction, mapping generation, evaluation and KG generation, each. The input modules focus on a data source sample, a related ontology, context enhancements, and prompt instructions for mapping generation. The mapping generation module can be configured with parameters for data size, ontology reduction method, and context enhancements. This module further orchestrates the processing steps, calls the ontology reduction module, and prepares the final prompt. Next, the generated mapping is evaluated against gold standard mappings which are created manually by experts. Finally, a KG generation module creates a KG based on the data and generated mapping. We conduct experiments with different configurations of *MYAM+R* across *three* real world data sets from Bosch. Naive ontology reduction and a mapping template as enhancement show promising as they achieve with column headers and first twenty data rows as input the best performance of *F1=0.74* for *YARRRML* and *F1=0.70* for *RML* mappings across all data sets. We achieve our objective in supporting experts in manufacturing mapping generation with the obtained results and feedback. Furthermore, our results present a baseline for future experiments in the manufacturing domain. We publish the source code[10] to support further research in this direction.

[10] `https://github.com/boschresearch/myamr_tgdk`

In future work, next to addressing the limitations discussed in Section 4, we enhance *MYAM+R* in several aspects. We will improve the implementation with ontology learning from data and matching with the given ontology. With this, we will evaluate *MYAM+R* on domain and upper ontologies. Further, it is promising to extend the input configuration with a representative sample from the data input. We will explore mapping quality improvements with multiple prompts and sequential chain approaches. With the intention to increase the level of support for mapping generation in manufacturing, we explore different research avenues, e.g., the inclusion of other LLMs as well as fine-tuned small language models for a given domain.

References

1 Bruno Charron, Yu Hirate, David Purcell, and Martin Rezk. Extracting semantic information for e-commerce. In Paul Groth, Elena Simperl, Alasdair J. G. Gray, Marta Sabou, Markus Krötzsch, Freddy Lécué, Fabian Flöck, and Yolanda Gil, editors, *ISWC*, volume 9982 of *Lecture Notes in Computer Science*, pages 273–290, 2016. doi:10.1007/978-3-319-46547-0_27.

2 Hanzhu Chen, Xu Shen, Qitan Lv, Jie Wang, Xiaoqi Ni, and Jieping Ye. Sac-kg: Exploiting large language models as skilled automatic constructors for domain knowledge graphs. *arXiv preprint arXiv:2410.02811*, 2024. doi:10.48550/arXiv.2410.02811.

3 Luciano Frontino de Medeiros, Freddy Priyatna, and Oscar Corcho. Mirror: Automatic R2RML mapping generation from relational databases. In *Engineering the Web in the Big Data Era: 15th International Conference, ICWE 2015, Rotterdam, The Netherlands, June 23-26, 2015, Proceedings 15*, pages 326–343. Springer, 2015. doi:10.1007/978-3-319-19890-3_21.

4 Anastasia Dimou, Dimitris Kontokostas, Markus Freudenberg, Ruben Verborgh, Jens Lehmann, Erik Mannens, Sebastian Hellmann, and Rik Van de Walle. Assessing and refining mappingsto RDF to improve dataset quality. In *The Semantic Web-ISWC 2015: 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II 14*, pages 133–149. Springer, 2015. doi:10.1007/978-3-319-25010-6_8.

5 Thomas Eiter, Tobias Geibinger, Nysret Musliu, Johannes Oetsch, Peter Skocovský, and Daria Stepanova. Answer-set programming for lexicographical makespan optimisation in parallel machine scheduling - ADDENDUM. *Theory Pract. Log. Program.*, 24(2):421, 2024. doi:10.1017/S1471068424000061.

6 Johannes Frey, Lars-Peter Meyer, Natanael Arndt, Felix Brei, and Kirill Bulert. Benchmarking the abilities of Large Language Models for RDF knowledge graph creation and comprehension: How well do LLMs speak Turtle? *arXiv preprint arXiv:2309.17122*, 2023. doi:10.48550/arXiv.2309.17122.

7 Johannes Frey, Lars-Peter Meyer, Felix Brei, Sabine Gründer-Fahrer, and Michael Martin. Assessing the evolution of LLM capabilities for knowledge graph engineering in 2023. In *ESWC*, volume 24, pages 26–30, 2024.

8 Irlán Grangel-González, Felix Lösch, and Anees ul Mehdi. Knowledge graphs for efficient integration and access of manufacturing data. In *25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 93–100, 2020. doi:10.1109/ETFA46521.2020.9212156.

9 Mohamed AG Hazber, Ruixuan Li, Guandong Xu, and Khaled M Alalayah. An approach for automatically generating R2RML-based direct mapping from relational databases. In *Social Computing: Second International Conference of Young Computer Scientists, Engineers and Educators, ICYCSEE 2016, Harbin, China, August 20-22, Proceedings, Part I 2*, pages 151–169. Springer, 2016.

10 Pieter Heyvaert, Ben De Meester, Anastasia Dimou, and Ruben Verborgh. Declarative rules for linked data generation at your fingertips! In *The Semantic Web: ESWC 2018 Satellite Events: ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers 15*, pages 213–217. Springer, 2018. doi:10.1007/978-3-319-98192-5_40.

11 Marvin Hofer, Johannes Frey, and Erhard Rahm. Towards self-configuring knowledge graph construction pipelines using llms-a case study with rml. In *Fifth International Workshop on Knowledge Graph Construction@ ESWC2024*, 2024.

12 Evgeny Kharlamov, Gulnar Mehdi, Ognjen Savkovic, Guohui Xiao, Elem Güzel Kalayci, and Mikhail 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.*, 56:11–29, 2019. doi:10.1016/J.WEBSEM.2018.10.004.

13 Andrew LeClair, Alicia Marinache, Haya El Ghalayini, Wendy MacCaull, and Ridha Khedri. A review on ontology modularization techniques - a multi-dimensional perspective. *IEEE Transactions on Knowledge and Data Engineering*, 35(5):4376–4394, 2023. doi:10.1109/TKDE.2022.3152928.

14 Benjamin Moreau and Patricia Serrano-Alvarado. Assessing the quality of RDF mappings with evamap. In *The Semantic Web: ESWC 2020 Satellite Events: ESWC 2020 Satellite Events, Heraklion, Crete, Greece, May 31–June 4, 2020, Revised Selected Papers 17*, pages 164–167. Springer, 2020. doi:10.1007/978-3-030-62327-2_28.

15 Jan Portisch, Michael Hladik, and Heiko Paulheim. Background knowledge in ontology match-

ing: A survey. *Semantic Web*, 15(6):2639–2693, 2024. doi:10.3233/SW-223085.

16 Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019. arXiv:1910.01108.

17 Patrick Sapel, Lina Molinas Comet, Iraklis Dimitriadis, Christian Hopmann, and Stefan Decker. A review and classification of manufacturing ontologies. *Journal of Intelligent Manufacturing*, 2024. doi:10.1007/s10845-024-02425-z.

18 Wilma Johanna Schmidt, Irlan Grangel-González, Tobias Huschle, Lena Wagner, Evgeny Kharlamov, and Adrian Paschke. Llm-supported mapping generation for semantic manufacturing treasure hunting. In *European Semantic Web Conference*, pages 84–101. Springer, 2025.

19 Wilma Johanna Schmidt, Irlan Grangel-González, Adrian Paschke, and Evgeny Kharlamov. MYAMR_TGDK. Software, partially funded by https://graph-massivizer.eu/, partially funded by https://www.smarty-project.eu/, partially funded by https://enrichmydata.eu/ (visited on 2025-12-08). URL: https://github.com/boschresearch/myamr_tgdk, doi:10.4230/artifacts.25262.

20 Wilma Johanna Schmidt, Diego Rincon-Yanez, Evgeny Kharlamov, and Adrian Paschke. Systematic Literature Review on Neuro-Symbolic AI in Knowledge Graph Construction for Manufacturing. *under review*, 2025.

21 Yuhu Shang, Yimeng Ren, Hao Peng, Yue Wang, Gang Wang, Zhong Cheng Li, Yangzhao Yang, and Yangyang Li. A perspective survey on industrial knowledge graphs: Recent advances, open challenges, and future directions. In *2023 International Conference on Machine Learning and Cybernetics (ICMLC)*, pages 194–200. IEEE, 2023. doi:10.1109/ICMLC58545.2023.10327989.

22 Zhigang Sun, Zixu Wang, Lavdim Halilaj, and Juergen Luettin. Semanticformer: Holistic and semantic traffic scene representation for trajectory prediction using knowledge graphs. *IEEE Robotics Autom. Lett.*, 9(9):7381–7388, 2024. doi:10.1109/LRA.2024.3426386.

23 Dylan Van Assche, Thomas Delva, Gerald Haesendonck, Pieter Heyvaert, Ben De Meester, and Anastasia Dimou. Declarative RDF graph generation from heterogeneous (semi-)structured data: A systematic literature review. *Journal of Web Semantics*, 75:100753, 2023. doi:10.1016/j.websem.2022.100753.

24 Dylan Van Assche, Thomas Delva, Pieter Heyvaert, Ben De Meester, and Anastasia Dimou. Towards a more human-friendly knowledge graph generation & publication. In *ISWC2021, The International Semantic Web Conference*, volume 2980. CEUR, 2021. URL: https://ceur-ws.org/Vol-2980/paper384.pdf.

25 Guohui Xiao, Diego Calvanese, Roman Kontchakov, Domenico Lembo, Antonella Poggi, Riccardo Rosati, and Michael Zakharyaschev. Ontology-based data access: A survey. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, 2018.

26 Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, and Soeren Auer. Quality assessment for linked data: A survey. *Semantic Web*, 7(1):63–93, 2016. doi:10.3233/SW-150175.

27 Xiang Zhang, Gong Cheng, and Yuzhong Qu. Ontology summarization based on rdf sentence graph. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 707–716, New York, NY, USA, 2007. Association for Computing Machinery. doi:10.1145/1242572.1242668.

28 Lingfeng Zhong, Jia Wu, Qian Li, Hao Peng, and Xindong Wu. A comprehensive survey on automatic knowledge graph construction. *ACM Computing Surveys*, 56(4):1–62, 2023. doi:10.1145/3618295.

29 Baifan Zhou, Yulia Svetashova, Andre Gusmao, Ahmet Soylu, Gong Cheng, Ralf Mikut, Arild Waaler, and Evgeny Kharlamov. Semml: Facilitating development of ML models for condition monitoring with semantics. *J. Web Semant.*, 71:100664, 2021. doi:10.1016/J.WEBSEM.2021.100664.