Towards self-configuring Knowledge Graph Construction Pipelines using LLMs - A Case Study with RML

Marvin Hofer[1,3], Johannes Frey[1,2] and Erhard Rahm[1,3]

[1]Institute of Computer Science, Leipzig University, Germany, https://cs.uni-leipzig.de
[2]KMI Competence Center @ Institute for Applied Informatics, Leipzig University, Germany, https://kmi-leipzig.de
[3]Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI), Dresden/Leipzig, Germany, https://scads.ai

Abstract

This paper explores using large language models (LLMs) to generate RDF mapping language (RML) files in the RDF turtle format as a key step towards self-configuring RDF knowledge graph construction pipelines. Our case study involves mapping a subset of the Internet Movie Database (IMDB) in JSON format given a target Movie ontology (selection of DBpedia Ontology OWL statements). We define and compute several scores to assess both the generated mapping files and the resulting graph using a manually created reference. Our findings demonstrate the promising potential of the state-of-the-art commercial LLMs in a zero-shot scenario.

Keywords

Knowledge Graph Construction, LLM-KG-Engineering, Automated RML Mapping Generation

1. Introduction

LLMs excel in understanding, generating, and manipulating human language. They are widely used for tasks like text completion, language translation, and generating creative content. Besides, LLMs have not only shown to be adept at natural language, but also to be capable of generating and understanding data representation languages [1] like the RDF Turtle format for RDF knowledge graphs (KGs). As such, LLMs demonstrated to be useful and evolve in assisting with knowledge graph engineering related task [2], including generation of RDF KGs from a variety of input formats ranging from textual to (semi)-structured sources [3, 4, 5]. However, when it comes to creating KGs out of (structured) sources, execution costs and runtime limit the scalibility and thus applicability of LLMs for transforming large input data into RDF. Currently, traditional mapping and transformation approaches scale much better in terms of costs and performance for structured sources. On the downside, most tools require a configuration of various input and output parameters (e.g. thresholds) per source, or even complex setups such as selecting scoring functions or defining rules, to work properly. As the complexity and variety of the input data or target KG grows, this effort becomes increasingly difficult [6]. We see

KGC

*Corresponding author.

hofer@informatik.uni-leipzig.de (M. Hofer); frey@informatik.uni-leipzig.de (J. Frey); rahm@informatik.uni-leipzig.de (E. Rahm)
0000-0003-4667-5743 (M. Hofer); 0000-0003-3127-0815 (J. Frey); orcid.org/0000-0002-2665-1114 (E. Rahm)

potential in combining the best of both worlds, by leveraging LLMs to automatically configure (the setup of) those tools, while scaling over both volume and variety.

In this work, we investigate such a hybrid strategy for the RDF Mapping language (RML) [7] as a proof of concept. Our case study is motivated and driven by the common scenario of mapping a semi-structured dataset into RDF for the purpose of integrating or fusing it into an existing target KG in an automated pipeline. RML is a powerful, and flexible language designed to enable the creation of RDF graphs from heterogeneous data sources, including JSON, XML, CSV, and relational databases. Given its popularity, standardization, RDF-compliant specification/declaration, and machine-readable language definition [8], it seems an excellent candidate for zero-shot experiments without prior training and fine-tuning, but also as a key technology towards self-configuring RDF KGC pipelines. Our work encompasses the following novelties and contributions:

- We conduct the first study on RML generation capabilities of LLMs
- We present novel insights into how well LLMs consider knowledge from target ontology definitions in Turtle when generating RML rules and selecting mapping targets
- We define and evaluate measures to analyze and evaluate LLM-generated RML mappings w.r.t. a gold standard reference KG.

The remainder of the paper is structured as follows: First, we will give a brief overview of works that use LLMs for data integration and knowledge graph construction. Then, we introduce our method and setup, describing our LLM instruction approach, the derived target ontology, and the test data snippets. Afterwards, we evaluate our approach using several commercial LLMs and investigate the quality with metrics involving a reference KG.

2. Related Work

2.1. LLMs in Mapping and KGC processes

A plethora of works experimented with the execution of KGC or data extraction / integration tasks in combination with or fully powered by LLMs. In [9] instruction prompts are used with large foundation models (released before the 2023 era, like GPT3) to perform entity matching, error detection, schema matching, data transformation, and data imputation tasks. Zhu et al. [3] investigated the performance of LLMs for KGC w.r.t. entity, relation, and event extraction as well as link prediction, on eight benchmark datasets. According to their study, LLMs showed limited effectiveness in a one to few-shot setting. SPIRES [4] recursively performs prompt interrogation to directly extract triples from text matching either a provided LinkML schema or identifiers from existing ontologies and vocabularies. In [1] a handful one-shot benchmark tasks are presented to evaluate the capabilities of LLMs (reported amongst others for GPT3.5/4, Claude 2) to parse, write, comprehend, fix and construct KGs in RDF Turtle Format. The performance of those commercial model version from July 2023 were reported as promising, motivating us to use the Turtle format for RML mappings and Ontology snippets.

TechGPT-2.0 [5] is a model trained specifically for KGC tasks, including named entity recognition and relationship triple extraction.

Also the field of Ontology Matching (OM) has seen several efforts to employ LLMs. OM is a crucial step to generate RML mappings between RDF KGs, but is also related to the task of mapping CSV or JSON schemata to a target ontology. [10] uses LLMs to refine ontology/vocabulary mappings. Similar to our experiment, they feed mapping candidates externally using high-recall methods (e.g. lexical similarity). Olala [11] also feeds textual descriptions of ontology candidate members into an LLM to perform binary or multiple-choice ontology matching decisions. AutoAlign [12] constructs a predicate-proximity-graph using LLMs to capture the similarity between predicates across two KGs. This allows for creating predicate embeddings without manual seeds and usage of unsupervised KG alignment in vector space without further use of LLMs.

Experiments in [13] indicate that LLMs seem able to understand and to construct concept hierarchies. We consider this an important skill in order to select appropriate classes and properties of a target ontology esp. when there are information granularity/depth mismatches.

A method that generates code using LLMs to create views on heterogeneous data lakes is proposed in [14].

While there exists a very recent effort to generate OWL ontologies and RML mappings for CSV files[1] with LLMs, the problem scenario and applied methods are different and more generic. The approach uses the LLM to create a novel target ontology based on the CSV structure, while we want to reuse an existing target ontology. Moreover, the RML generation is constrained by pre-defined generic RML snippet templates, whereas the LLM needs to compose the final mapping by creating instances of those templates and composing them in one file. To the best of our knowledge, no evaluation for creating RML mappings for a JSON dataset without further assistance - other than providing the relevant target ontology members - has been conducted.

2.2. Mapping Frameworks and RML Quality Evaluation

Several frameworks and mapping languages exist for (declarative) graph generation from heterogeneous (semi-)structured data. Since we focus on RML [7] and RMLMapper in this work, we refer the reader to [15] for an overview on other approaches. With regard to evaluating the quality of RML mappings, a few works exist. Most notably, [16] created a tool[2] that assesses RML mappings and suggests patches. Backed by RDFUnit (tests) and OWL axioms from the target ontology, the RML mapping file can be checked for quality issues (e.g. domain/range violations, missing type statements). The approach was evaluated, amongst others, on the DBpedia dataset for both mappings and mapped data.

Four metrics to assess the quality of R2RML (a subset of RML) mappings were presented in [17]. The tests checked for usage of undefined classes/properties, blank nodes and RDF reification as indicators for quality issues. In [18] this was extended into a framework that uses SHACL to check (e.g. similar to [16] the correct term type and datatype) and refine R2RML mappings. The novel RML ontology [8] is shipped with SHACL shapes, that could be used to validate proper usage of RML statements in the mappings. EvaMap [19] is a framework to evaluate RDF Mappings using a set of metrics based on the 7 Linked Data quality dimensions from [20]. The metrics are evaluated given the input dataset on a YARRRML mapping [21] or sampled mapped entities (when instances are needed) and aggregated into a final weighted score. Notably is that they measure the mapping coverage w.r.t. the input dataset as well as

[1] https://github.com/tecnomod-um/OntoGenix

[2] https://github.com/RMLio/RML-Validator

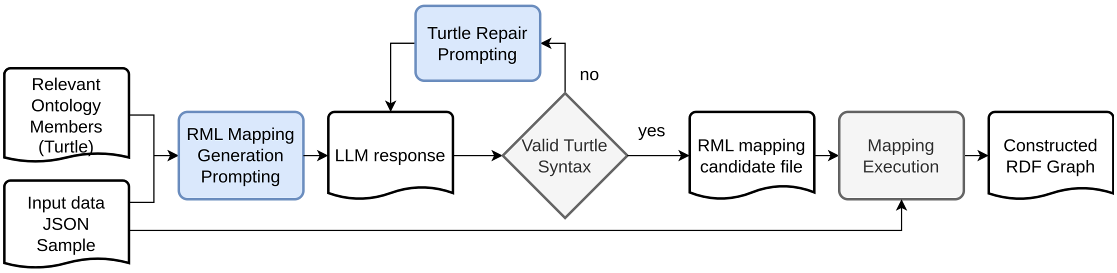


Figure 1: Mapping Generation Experiment Workflow.

[22] which calculate the ratio of mapped SQL columns in relational database mapping scenarios. [22] provides an exhaustive list of measures for faithfulness, quality, and interoperability of the output data. However, a plethora of them is out of scope or do not apply to our work, since they require knowledge from the data(base) in use or assume publishing a dataset given an open vocabulary choice.

Although some of our evaluation scores can be grounded in these previous works, we did not directly employ any of those tools. Since we consider quality mostly as fitness for use and given the differences in requirements and the nature of the problem setups, we need a custom set of scores for more nuanced insights.

3. Experiment Method and Setup

In this section, we describe the data and ontology used in our experiment, the prompts as well as requirements and challenges w.r.t. the RML generation task. Fig. 1 gives an overview on the experimental mapping generation workflow. We executed RML mappings using one of the reference implementations in Java[3]. The code and all resources are accessible in a public repository [4], including (links to) the input data, the crafted prompt templates, and generated results with their descriptions.

3.1. Test Data & Prompt Input Data

We derived our test data from the International Movie Database (IMDB)[5]. We convert a subset (focused on movies and involved Persons) of the original CSV files into JSON (see Listing 1) format as input data for the LLM. This is done because processing the CSV export of a relational table adds several layers of complexity. First, the data is split into various files. Second, it has multi-value cells (`'|'` separator) and empty cells (`'\N'` values). Third, CSV values have no datatype information. These introduce extra sub-tasks with potential error sources, such as CSV syntax analysis or advanced RML expressions (e.g., string splitting). However, we want to start with assessing the general capability on how well LLMs can generate RML mappings having as much information as clearly and "syntactically standardized" as possible at hand. JSON on

[3] https://github.com/RMLio/rmlmapper-java
[4] https://github.com/Vehnem/kg-pipeline/tree/main/experiments/llm4rml
[5] https://developer.imdb.com/non-commercial-datasets/

```
{
  "id": "tt0167423",
  "originalTitle" : "Diamonds",
  "runtimeMinutes" : 91,
  "startYear" : 1999,
  "genre" : ["Comedy","Mistery"],
  "titleTyp" : "movie",
  "isAdult" : 0,
  "involvedPeople" : [{
    "id" : "tt0167423",
    "ordering" : 1,
    "name" : "Kirk Douglas"
    "birthYear" : 1916,
    "deathYear" : 2020,
    "category" : "actor" }, ...]
}
```

Listing 1: JSON Input Data Excerpt.

```
# @prefix ...
@base <http://mykg.org/resource/>

<tt0167423> a dbo:Film ;
  dbo:title "Diamonds" ;
  dbo:genre "Comedy", "Mistery" ;
  dbo:startYear "1999"^^xsd:gYear ;
  dbo:Work/runtime "91"^^dtd:minute ;
  dbo:starring <nm0000018> , ... ;
  dbo:director <nm0038875> ;
  ...

<nm0000018> a dbo:Person ;
  dbo:name "Kirk Douglas" ;
  dbo:birthYear "1916"^^xsd:gYear ;
  dbo:deathYear "2020"^^xsd:gYear .
```

Listing 2: Gold Reference Snippet.

the other hand is a popular representation format used in many web applications / APIs. A vast amount of JSON data is available in corpora used to train LLMs. It allows to define nested values, such as people associated with films, and arrays to represent multiple values, such as film genres. It also supports standard datatypes like integers and floats. Nevertheless, we see comparing the performance between JSON and the original CSV as interesting future work. We select a single JSON sample by filtering the title to cover a wide range of specified fields, including all possible people job categories specified in our target Movie ontology. Listing 1 shows a snippet from our selected sample, and Listing 2 the respective data in RDF Turtle.

In our final experiment, we will test a single film resource, about the movie "Diamonds". The film has connections to 10 individuals, covering all possible IMDB job categories. We expect the derived RDF graph to contain 59 triples based on 15 mapped properties.

3.2. Target Ontology

The target ontology (see Fig. 2) is a subset of existing classes and properties from the DBpedia ontology by focusing on films and involved people. We compiled, to the best of our knowledge, the best matching concepts relevant to our input data, together with context from the ontology (super-class, domain/range etc). But we also added real-life "confusables": two `runtime` properties (one assuming minutes the other seconds as double), (film) `editing` vs. `editor` (publishing) properties, `producer` vs `executive producer` properties. The ontology consists of 6 classes (Work, Film, VideoGame, Person, Actor, Agent). As a special minor change, we modified the property `dbo:genre`, and defined it as `owl:DatatypeProperty` instead of `owl:ObjectProperty` as the latter would require the LLM to construct new genre entities from a simple string or matching them to DBpedia entities. Both tasks introduce error sources that affect the evaluation of basic RML generation capabilities. We consider them as a subsequent step in a KGC pipeline, once an initial KG version of the input was created via the

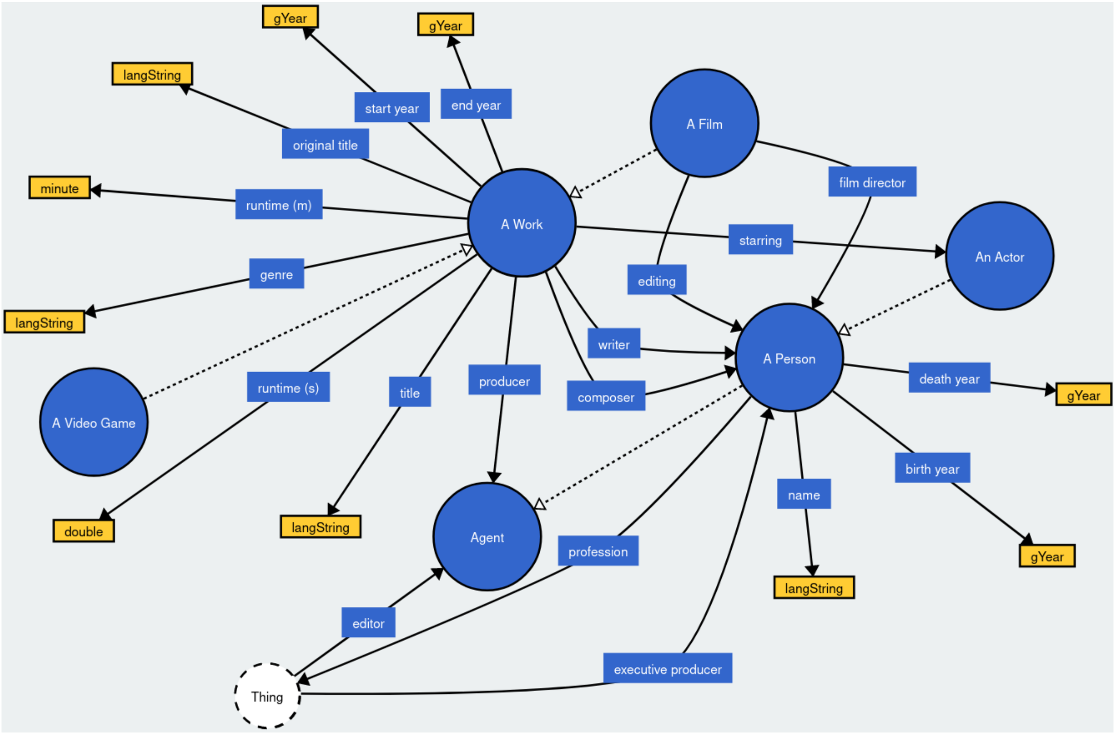


Figure 2: Target Ontology about Movies.

mapping, and thus out of the scope of this initial experiment. Although the target ontology is rather specific, it serves the purpose to check whether the LLms are capable of using it correctly.

3.3. RML Mapping Requirements & Challenges

In order to generate (meaningful) data for our use case of constructing a KG to be fused with a target KG, the generated mapping has to fulfill a set of requirements:

1. defining correct logical source based on given file path
2. mapping all JSON attributes where a target property (candidate) exists in the ontology but not mapping keys without a candidate (ontology coverage & succinctness)
3. selecting most specific over generic properties and types (e.g. Actor instead of Person) w.r.t. formalized context of ontology members
4. correct literal value representations and datatypes
5. following a specified pattern for entity IRIs incorporating their IDs
6. usage of RML-Mapper built-in functions only

Given these requirements and our experiment data, we consider mapping the category attribute of involved persons to the appropriate job function/role property of the ontology a challenge. For example, when the category of an involved person is *actor* or *actress* it needs to generate an edge from the movie entity via *dbo:starring* to the actor entity. Another interesting case is posed by both runtime properties, whereas one needs conversion from minutes to seconds.

3.4. LLM Instructions

Prompt engineering is an iterative process. We started by crafting simple prompts that quickly accomplished our tasks and then refined them based on our early findings.

3.4.1. RML Mapping Prompt

The mapping file generation prompt encompasses different rules and presets the entire target ontology and a JSON input data sample. The instruction contains the following rules (shortened versions):

- Convert given JSON data with file located at /path/to/input.json
- Use the provided movie ontology as a mapping target
- Map information with the most specific class or property possible
- Take the domain and range of properties into account
- Convert values to be valid for datatypes
- Ensure syntactical and semantical correctness to RML specifcation
- Use 'http://mykg.org/resource/' as target namespace

3.4.2. Turtle Repair Prompt

One major challenge when requesting specific structured formats from LLMs in a zero-shot manner is the handling of syntax errors.

To deal with corrupted syntax in the Turtle format, we use a second prompt for requesting the LLM to repair the syntax. It includes rigorous instructions and hints for the syntax, the corrupted Turtle data (response from initial mapping generation prompt or the output of previous fixing attempt), and the parsing exception thrown by the Python RDF library (rdflib).

The instruction contains the following constraints (shortened versions):

- Respond with the full fixed RDF Turtle document.
- Stick with the original structure and formatting of the original Turtle file as much as possible.
- Only apply minor modifications to fix the syntax.
- Take the given parsing exception into account when repairing
- Check proper usage of `.` `;` `,` for separating triples, predicate-objects, and objects.

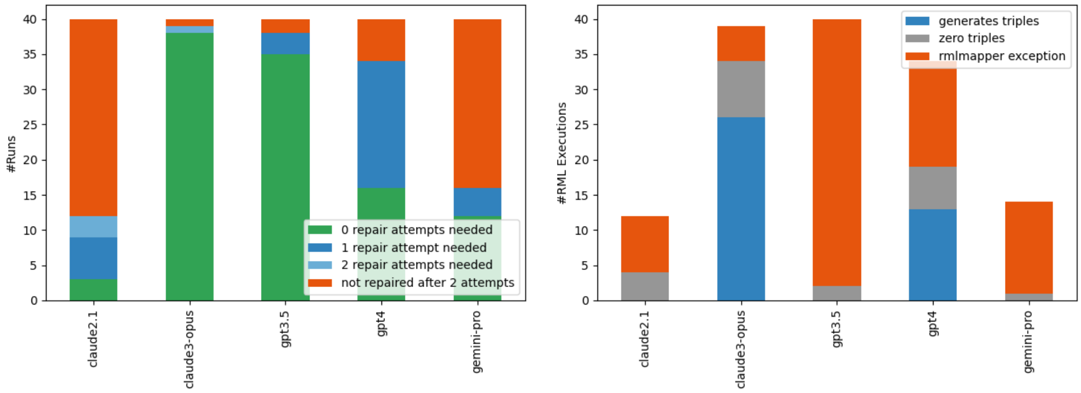
4. Evaluation

In our evaluation, we assessed five commercial large language models (LLMs): Claude2.1, Claude3 Opus (20240229), GPT3.5 Turbo (01-25), GPT4 Turbo (01-25-preview), and Gemini Pro. Each model underwent a structured testing protocol over 40 runs using its default settings, such as temperature. The evaluation comprised the following steps:

1. For each run, we checked the output for RDF syntax errors. If the initial output is not syntactically valid, we performed up to two consecutive repair attempts.
2. For outputs that were either generated correctly or successfully repaired, we evaluated the generation of triples.

3. We then verified the correctness of these triples, both under exact and relaxed conditions.
4. Finally, for all properties, we assessed whether they were correctly mapped to the target ontology.

We assess the generated RML and the transformed RDF triples in the following sections.



(a) RML Turtle Syntax Validity

(b) Mapping Soundness

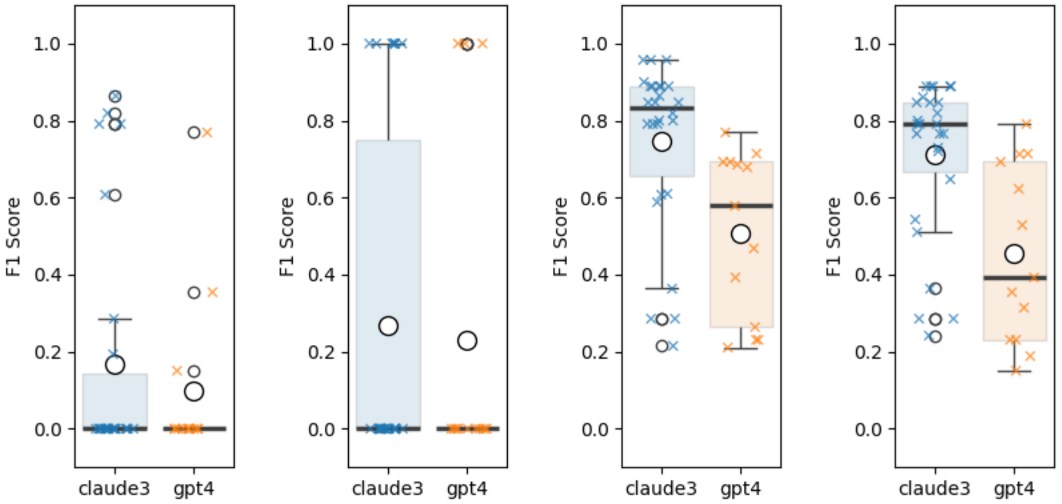
Figure 3: LLM Response Validity of generated RML Mapping Files in Turtle Format.

4.1. LLM Response Validity

In Fig. 3a, we show how many iterative Turtle syntax repair attempts were performed for each of the 40 generated initial mappings per model. We found, that the novel Claude3 is the most syntax-conformant model and shows a significant enhancement over its predecessor (which lacks in generating and repairing Turtle in around 75% of the cases). For the OpenAI models we could observe a reversed effect, which aligns with findings reported in [1]. Fortunately, GPT4 can fix its mappings in the majority of cases in one attempt. Gemini has a slightly better performance (25%) than Claude2.1. However, when looking at Fig. 3b, we see that when trying to put the valid (parsable) mappings into action, both models do not generate a single mapping that produces any triple but the majority leads to processing exceptions (correct syntax but issues w.r.t. RML spec or bad JSON iterator). While GPT3.5 is reliable w.r.t. Turtle, the produced RML is not sound and leads to errors in over 90% of the cases. Claude3 produces the most actionable mappings in the majority of cases, where a great portion generates triples. GPT4 only creates mappings in slightly more than one-third of the cases. The reason for 0-triple mappings is usually that the iterators or reference conditions do not select input data.

4.2. Mapping Quality.

We analyze the quality of the mapping mostly based on the "fitness for use" of the triples generated by it. Fitness for use is driven by the degree of how well requirements 2-5 (Section 3.3) are fulfilled and our data integration/fusion use case (mapping data to integrate in target KG) can be accommodated. While we reused some metrics from related work, a plethora of these do



(a) Triples (b) Subjects (c) Predicates (d) Objects

Figure 4: Generated Knowledge Graph Elements Quality. Exact Match on Triple and Triple Elements. The F1 scores are presented, with the median (bold line) and mean (big circle) scores.

	claude3	gpt4
mappings with triples	**26**	**13**
full predicate coverage	4	0
only ontology mapped	20	2
isAdult|ordering not mapped	26	13
usage of any / custom funct.	0/0	3/3

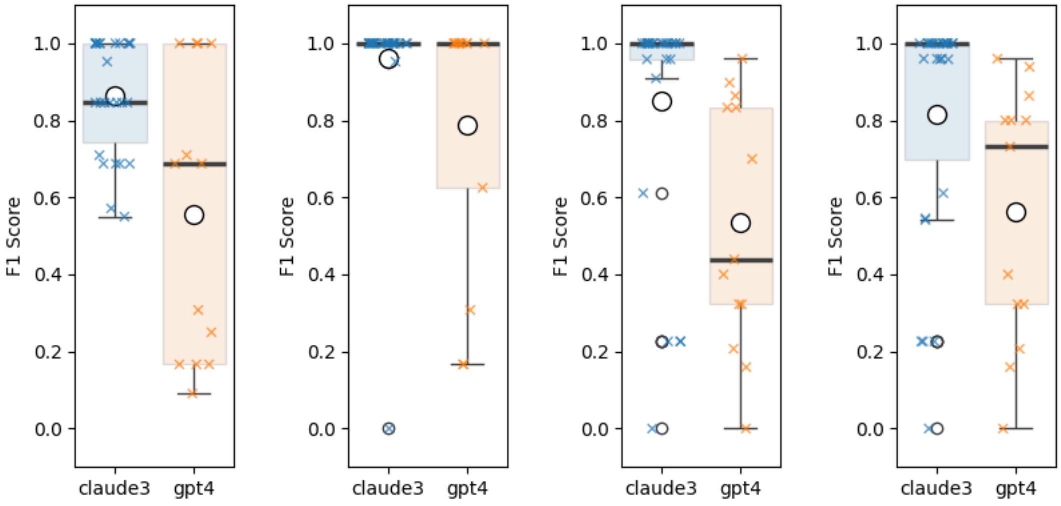
Table 1
Vocabulary Usage.

	claude3	gpt4
mappings with triples	**26**	**13**
all people have id	21	9
all people ids are typed	20	7
all actors have id	21	9
all actor ids are typed	11	0

Table 2
Generated Entity IRIs and Types.

not match well with this setting or have limited value. E.g. correctness of the (lexical) values of transformed data is more important than the datatype (ideally datatype is also correct but if the value is wrong and datatype is correct this a serious problem). As such using e.g. RDFUnit as in [16] would be of limited help. Instead, we created a "gold mapping" and produce a gold standard KG. However, matching the mapping output KGs with it, pose the challenge that the graph could be correct and useful but not isomorphic due to IRI identifier variety. Moreover, AI-generated mappings can have different types of errors compared to manually created ones. But this work has the goal to get insights what aspects of the experiment task pose challenges to the LLMs. As a consequence we present a set of custom measures that have varying levels of strictness, when it comes to comparing the triples generated by the mapping to the gold standard, but also measures that are tailored to our specific experiment data. In Table 1 we report overall statistics about the mappings and its vocabulary usage. Claude3 created 26 and GPT4 created 13 mappings that generated at least one triple. With regard to mapping to all possible candidates from the target ontology, only four Claude3 mappings succeeded. Claude3 also matched in most cases only the correct candidates and nothing more. For GPT4 this OM aspect seems to cause problems. It consistently tried to map "unmappable" elements like *isAdult* or *ordering*. Moreover, it also ignored our requirement to use no external functions. We found that GPT4 has issues in following our mapping requirements and instructions.

4.2.1. Triples Exact Match Comparison



(a) Class Assignment (b) Subject Relaxed (c) Literal Values EM (d) Literal Datatypes EM

Figure 5: Relaxed Scores.

As a first and very strict set of scores we report 4 graph identity measures. In Fig. 4, we show F1 scores for the extracted triples compared to our gold reference. Every point of the boxplot represents a mapping that generated at least one triple. Note, that the amount of mappings are different because, we gave every model the same fair chance in 40 runs to produce and repair the mapping, which lead to different numbers of mappings generating triples. The median (bold line) for the triple-oriented and subject-oriented matches is both 0. The mean (big circle) of Claude3 is slightly better than GPT4. However, we also see some outliers with F1 values of 0.8 which show that some runs were more successful. With regard to the set of predicates and objects the results are significantly better, showing that Claude3 performed better.

4.2.2. Relaxed Scores

The triple-element scores showed that the F1 triple score is significantly affected by problems with subject identifiers. As such, we present in Fig. 5 a set of relaxed scores. The **Subject Relaxed** score is tolerant with regard to our requested IRI pattern (since the models very often ignore this request and create IRIs of the form /class/id instead). Since the IRI patterns have no effect on data integration, we tolerate this in the relaxed score, but we ensure that the correct

	type	Work/runtime	birthYear	composer	deathYear	director	editing	executiveProducer	genre	name	originalTitle	starring	startYear	title	writer	editor	producer	runtime
p is used	26	24	25	6	25	6	5	0	22	23	25	7	25	24	6	1	6	0
p outdegree OK	7	24	22	5	22	5	5	0	22	20	25	4	25	24	5	-	-	-
s-o fuzzy OK	9	20	21	0	21	0	0	0	18	20	21	0	21	20	0	-	-	-
o is IRI	26	0	0	6	0	6	5	0	1	1	1	7	0	1	6	1	6	0
o is literal	0	24	25	0	25	0	0	0	21	22	24	0	25	23	0	0	0	0
literal OK	-	19	21	-	21	-	-	-	18	20	21	-	21	20	-	-	-	-

Table 3
Claude 3 Property Mapping Statistics. Showing counts from a **total of 26 generated** files with triples. The last three rows show the correct mapping of object properties "o is IRI", and datatype properties "o is literal", including datatype declaration "literal OK".

ids are contained within the IRIs, since this a non-negotiable necessary requirement. This score shows that Claude3 makes only very few errors w.r.t. subject id generation, but also GPT4 has a reasonable performance. Since the object performance can be also affected by our IRI constraint for Object properties, we had an isolated look on the datatype properties. The **Literal Values EM** score describes whether the values (lexical) are having an exact match without considering the datatype. We see that Claude3 shows an excellent performance compared to GPT4 (median smaller than 0.5). When doing the opposite by ignoring the object value but testing for the correct datatype only (**Literal Datatypes EM**), the performance of GPT4 is better, but still worse compared to Claude3. We calculated a **Class Assignment** score to get an impression of which entities are mapped. The class assignment result shows the F1 score calculated between the list of expected (most specific) and assigned class types of the applied RML mapping. We see that Claude3 is better but the assignment needs further inspection. In Table 2, we see both Claude3 and GPT4 miss in some instances to extract all person entities but then in particular fail to assign the correct most-specific type to Actors.

4.2.3. Property Mapping Insights

Based on our analysis of the single mappings, we have observed diverse vocabulary usage throughout the generated mapping files. Table 3 and Table 4 show the results of an in-depth analysis for Claude3 and GPT4. We analyze the **mapping vocabulary usage quality** by: counting the number of generated triple files containing the property ("p is used"); checking whether the outdegree matches the outdegree from the reference ("p outdegree OK"); checking whether it is used reasonably ("s-o fuzzy OK") by performing fuzzy matching on subjects (should contain title/person id) and objects (match on value ignoring the datatype) with the reference triples; it is used as an object or datatype property, and in the case of the latter, whether the literal value and datatype match correctly ("literal OK").

We observed that all **datatype properties** are highly used in the generated mappings by both

	type	Work/runtime	birthYear	composer	deathYear	director	editing	executiveProducer	genre	name	originalTitle	starring	startYear	title	writer	editor	producer	runtime
p is used	13	1	8	1	8	3	0	0	11	9	10	2	11	11	1	1	1	10
p outdegree OK	0	1	7	0	7	1	0	0	11	8	10	0	11	11	0	-	-	-
s-o fuzzy OK	0	0	8	0	8	0	0	0	10	8	9	0	10	9	0	-	-	-
o is IRI	13	0	0	0	0	2	0	0	1	0	0	1	0	0	0	0	0	0
o is literal	0	1	8	1	8	1	0	0	10	9	10	1	11	11	1	1	1	10
literal OK	-	-	8	-	8	-	-	-	10	8	9	-	10	9	-	-	-	-

Table 4
GPT 4 Property Mapping Statistics. Showing counts from a **total of 13 generated** files with triples.

models. Considering a fuzzy triple match ("s-o fuzzy OK") for triples containing the predicate shows good usage counts for each of these properties. The same can be seen for property outdegree ("p outdegree OK") and their value+datatype correctness count ("literal OK"). Claude3 has at least 18, and GPT4 has at least 8 generated triple files with datatype properties, mapping the correct value and datatype. Concerning the "confusable" predicates (Section 3.3), our analysis shows that (incorrectly) mapping to the property `runtime` instead of `Work/runtime` only happened for GPT4.

Both models fail to generate correct mapping rules for all the job function **object properties** based on the given person (involvement/job) category. Claude3 and GPT4 do not map the expected property `executiveProducer`, but the `producer` property was mapped six times by Claude3 and once by GPT4. Further, a mapping for the false property `editor` was only generated once by both models. However, Claude3 RML mappings use the expected target property `editing` five times, but unfortunately incorrectly, whereas GPT4's mapping results do not contain a single triple using this property.

Notably, for both models, we can identify the **incorrect usage of datatype or object properties** in several cases (e.g., an object is mapped as literal instead of IRI, or vice versa). The genre property was used as object property once in each of the model results, as opposed to the changes made in our ontology (w.r.t. the original definition in the DBpedia ontology). The highest incorrect usage in both models is for the job function properties (e.g., `starring`).

5. Conclusion & Future Work

In conclusion, our detailed analysis of RML mappings generated by the latest commercial large language models (LLMs) like Claude3 Opus and GPT4 showcased promising results in handling given JSON data and target ontologies. Claude3 Opus, in particular, generally excelled over GPT4 in quality evaluations, highlighting its enhanced proficiency. However, earlier versions such as Claude 2.1, GPT3.5T, and Gemini-Pro struggled significantly with the task, failing to produce any valid RML documents.

Throughout our development phase, we encountered issues related to RDF syntax, such as incorrect prefix definitions and the use of separators in Turtle syntax. Although the newer models showed a good grasp of the RML spec and our ontology vocabulary, they consistently confused properties like executive-producer with producer, likely due to unclear ontology documentation, and incorrectly mapped certain properties completely (person job functions). The property mapping issue may be related to the correct definition of the JSON reference, whereby The LLM needs to generate a JSON expression for each job, matching an exact key-value pair in the nested person objects, to map the correct specific role function property.

Our initial experiment offers plenty of directions for future research. Testing RML mapping with different data formats, such as CSV and XML, could offer insights into how data expressiveness impacts mapping generation. The study of YARRRML could further our understanding of LLM capabilities in relation to the targeted mapping issues. Next, experimenting with diverse data sampling strategies, like popular versus unpopular datasets or entirely unknown data, would measure understanding of more specialized domains. Furthermore, enhancing model performance through feedback mechanisms on errors and the integration of SHACL validations could improve results. There is also a potential to refine LLM capabilities by providing more context through ontologies and improving model training with existing mappings from resources like GitHub. Finally, enabling custom RML function usage and generation would offer better flexibility and allow more complex mappings.

Moving forward, we see substantial potential for LLMs to automate and enhance the construction of knowledge graphs, highlighting the importance of continued advancements and refinements in hybrid KG and LLM technologies.

Acknowledgements. The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany and by the Sächsische Staatsministerium für Wissenschaft Kultur und Tourismus in the program Center of Excellence for AI-research "Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig", project identification number: ScaDS.AI.

Furthermore, this work was partially supported by grants from the German Federal Ministry for Economic Affairs and Climate Action (BMWK) to the KISS project (01MK22001A).

References

[1] J. Frey, L.-P. Meyer, N. Arndt, F. Brei, K. Bulert, Benchmarking the abilities of large language models for RDF knowledge graph creation and comprehension: How well do llms speak turtle?, in: M. Alam, M. Cochez (Eds.), Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG 2023) co-located with the 21th International Semantic Web Conference (ISWC 2023), Athens, November 6-10, 2023, volume 3559 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3559/paper-3.pdf. arXiv:2309.17122.

[2] J. Frey, L.-P. Meyer, F. Brei, S. Gründer-Fahrer, M. Martin, Assessing the evolution of llm capabilities for knowledge graph engineering in 2023, in: ESWC 2024 Satellite Events, Hersonissos, Crete, Greece, May 26 - 30, 2024, Proceedings., 2024.

URL: https://www.researchgate.net/publication/378804553_Assessing_the_Evolution_of_LLM_capabilities_for_Knowledge_Graph_Engineering_in_2023.

[3] Y. Zhu, X. Wang, J. Chen, S. Qiao, Y. Ou, Y. Yao, S. Deng, H. Chen, N. Zhang, Llms for knowledge graph construction and reasoning: Recent capabilities and future opportunities, 2023. arXiv:2305.13168.

[4] J. H. Caufield, H. Hegde, V. Emonet, N. L. Harris, M. P. Joachimiak, N. Matentzoglu, H. Kim, S. A. T. Moxon, J. T. Reese, M. A. Haendel, P. N. Robinson, C. J. Mungall, Structured prompt interrogation and recursive extraction of semantics (spires): A method for populating knowledge bases using zero-shot learning, 2023. arXiv:2304.02711.

[5] J. Wang, Y. Chang, Z. Li, N. An, Q. Ma, L. Hei, H. Luo, Y. Lu, F. Ren, Techgpt-2.0: A large language model project to solve the task of knowledge graph construction, 2024. arXiv:2401.04507.

[6] M. Hofer, D. Obraczka, A. Saeedi, H. Köpcke, E. Rahm, Construction of knowledge graphs: State and challenges, CoRR abs/2302.11509 (2023). URL: https://doi.org/10.48550/arXiv.2302.11509. doi:10.48550/ARXIV.2302.11509. arXiv:2302.11509.

[7] A. Dimou, M. V. Sande, P. Colpaert, R. Verborgh, E. Mannens, R. V. de Walle, RML: A generic language for integrated RDF mappings of heterogeneous data, in: C. Bizer, T. Heath, S. Auer, T. Berners-Lee (Eds.), Proceedings of the Workshop on Linked Data on the Web co-located with the 23rd International World Wide Web Conference (WWW 2014), Seoul, Korea, April 8, 2014, volume 1184 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2014. URL: https://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf.

[8] A. Iglesias-Molina, D. Van Assche, J. Arenas-Guerrero, B. De Meester, C. Debruyne, S. Jozashoori, P. Maria, F. Michel, D. Chaves-Fraga, A. Dimou, The rml ontology: A community-driven modular redesign after a decade of experience in mapping heterogeneous data to rdf, in: T. R. Payne, V. Presutti, G. Qi, M. Poveda-Villalón, G. Stoilos, L. Hollink, Z. Kaoudi, G. Cheng, J. Li (Eds.), The Semantic Web – ISWC 2023, Springer Nature Switzerland, Cham, 2023, pp. 152–175.

[9] A. Narayan, I. Chami, L. Orr, S. Arora, C. Ré, Can Foundation Models Wrangle Your Data?, 2022. URL: http://arxiv.org/abs/2205.09911, arXiv:2205.09911 [cs].

[10] N. Matentzoglu, J. H. Caufield, H. B. Hegde, J. T. Reese, S. Moxon, H. Kim, N. L. Harris, M. A. Haendel, C. J. Mungall, Mappergpt: Large language models for linking and mapping entities, 2023. arXiv:2310.03666.

[11] S. Hertling, H. Paulheim, Olala: Ontology matching with large language models, in: Proceedings of the 12th Knowledge Capture Conference 2023, K-CAP '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 131–139. URL: https://doi.org/10.1145/3587259.3627571. doi:10.1145/3587259.3627571.

[12] R. Zhang, Y. Su, B. D. Trisedya, X. Zhao, M. Yang, H. Cheng, J. Qi, AutoAlign: Fully Automatic and Effective Knowledge Graph Alignment enabled by Large Language Models, 2023. URL: http://arxiv.org/abs/2307.11772, arXiv:2307.11772 [cs].

[13] M. Funk, S. Hosemann, J. C. Jung, C. Lutz, Towards ontology construction with language models, in: S. Razniewski, J. Kalo, S. Singhania, J. Z. Pan (Eds.), Joint proceedings of the 1st workshop on Knowledge Base Construction from Pre-Trained Language Models (KBC-LM) and the 2nd challenge on Language Models for Knowledge Base Construction (LM-KBC) co-located with the 22nd International Semantic Web Conference (ISWC 2023), Athens,

Greece, November 6, 2023, volume 3577 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3577/paper16.pdf.

[14] S. Arora, B. Yang, S. Eyuboglu, A. Narayan, A. Hojel, I. Trummer, C. Ré, Language models enable simple systems for generating structured views of heterogeneous data lakes, Proc. VLDB Endow. 17 (2023) 92–105. URL: https://doi.org/10.14778/3626292.3626294. doi:10.14778/3626292.3626294.

[15] D. Van Assche, T. Delva, G. Haesendonck, P. Heyvaert, B. De Meester, A. Dimou, Declarative rdf graph generation from heterogeneous (semi-)structured data: A systematic literature review, Journal of Web Semantics 75 (2023) 100753. URL: https://www.sciencedirect.com/science/article/pii/S1570826822000373. doi:https://doi.org/10.1016/j.websem.2022.100753.

[16] A. Dimou, D. Kontokostas, M. Freudenberg, R. Verborgh, J. Lehmann, E. Mannens, S. Hellmann, R. Van de Walle, Assessing and refining mappingsto rdf to improve dataset quality, in: M. Arenas, O. Corcho, E. Simperl, M. Strohmaier, M. d'Aquin, K. Srinivas, P. Groth, M. Dumontier, J. Heflin, K. Thirunarayan, S. Staab (Eds.), The Semantic Web - ISWC 2015, Springer International Publishing, Cham, 2015, pp. 133–149.

[17] A. Randles, D. O'Sullivan, Assessing quality of R2RML mappings for osi's linked open data portal (short paper), in: B. Yaman, M. A. Sherif, A. N. Ngomo, A. Haller (Eds.), Proceedings of the 4th International Workshop on Geospatial Linked Data (GeoLD) Co-located with the 18th Extended Semantic Web Conference (ESWC 2021), Virtual event (instead of Hersonissos, Greece), June 7th, 2021, volume 2977 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021, pp. 51–58. URL: https://ceur-ws.org/Vol-2977/paper7.pdf.

[18] A. Randles, A. C. Junior, D. O'Sullivan, A framework for assessing and refining the quality of r2rml mappings, in: Proceedings of the 22nd International Conference on Information Integration and Web-Based Applications & Services, iiWAS '20, Association for Computing Machinery, New York, NY, USA, 2021, p. 347–351. URL: https://doi.org/10.1145/3428757.3429089. doi:10.1145/3428757.3429089.

[19] B. Moreau, P. Serrano-Alvarado, Assessing the quality of rdf mappings with evamap, in: A. Harth, V. Presutti, R. Troncy, M. Acosta, A. Polleres, J. D. Fernández, J. Xavier Parreira, O. Hartig, K. Hose, M. Cochez (Eds.), The Semantic Web: ESWC 2020 Satellite Events, Springer International Publishing, Cham, 2020, pp. 164–167.

[20] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann, S. Auer, Quality assessment for linked data: A survey, Semantic Web 7 (2015) 63–93. doi:10.3233/SW-150175.

[21] P. Heyvaert, B. D. Meester, A. Dimou, R. Verborgh, Declarative rules for linked data generation at your fingertips!, in: A. Gangemi, A. L. Gentile, A. G. Nuzzolese, S. Rudolph, M. Maleshkova, H. Paulheim, J. Z. Pan, M. Alam (Eds.), The Semantic Web: ESWC 2018 Satellite Events - ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers, volume 11155 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 213–217. URL: https://doi.org/10.1007/978-3-319-98192-5_40. doi:10.1007/978-3-319-98192-5_40.

[22] D. Tarasowa, C. Lange, S. Auer, Measuring the quality of relational-to-rdf mappings, in: P. Klinov, D. Mouromtsev (Eds.), Knowledge Engineering and Semantic Web, Springer International Publishing, Cham, 2015, pp. 210–224.