Accessing Semi-structured Data with RML and LLMs (Extended Abstract)

Shikhat Karkee[1], Elena Botoeva[1], Sam Coombes[2], Anna Jordanous[1], Özgür Kafali[1] and Davide Lanti[3]

[1]University of Kent, Canterbury, UK
[2]East Kent Hospitals University Foundation Trust, Ashford, UK
[3]Free University of Bozen-Bolzano, Bolzano, Italy

Abstract

We propose a methodology for generating knowledge graphs from semi-structured data that integrates large language model (LLM) queries directly into RML mapping rules: structured fields are handled by standard RML rules, while LLMs extract ontological terms from unstructured text fields. This targeted use of LLMs–fed only short, focused text snippets–reduces the risk of errors compared to end-to-end LLM solutions, while preserving correctness guarantees for the structured part of the data. We evaluate our approach on a Medicines Information use-case using pharmaceutical data in JSON format, and assess the correctness of the resulting knowledge graphs and query answers against pure LLM-based baseline.

This is an extended abstract of the paper published at DL 2025 titled "Accessing Semi-structured Data with RML and LLMs" https://ceur-ws.org/Vol-4091/paper54.pdf.

Keywords

Semi-structured data, Large Language Models, Knowledge Graphs, RML

1. Introduction

The Ontology-Based Data Access (OBDA) framework [1] has provided the theoretical foundations for representing data stored in a database as a *knowledge graph (KG)* – a collection of facts formulated using high-level concepts and relationships. KGs can be queried using formal languages such as SPARQL [2], the standard query language for RDF [3]. Importantly, for such *structured* data representations and query languages, there is a guarantee that the answers are correct as long as the data is correct.

A crucial component of the OBDA framework is *mappings*—declarative specifications of how KG entities are populated from data values. For instance, consider the following mapping rules (in an abstract syntax) with a source SQL query and two target RDF triple templates:

```
source    SELECT m_id, name  FROM medicines
target    :med-{m_id}  rdf:type  :Drug .
          :med-{m_id}  :prefLabel  {name} .
```

Executing them would, for each row in the `medicines` relation, create an instance of the `:Drug` class, identified by an IRI instantiated from the template `:med-{m_id}` and labelled (property `:prefLabel`) with the value of attribute `name`. R2RML is the standard mapping language from relational databases to RDF [4], while RML is its extension to generic data sources, such as JSON, CSV and XML [5, 6]. RML mappings can be executed by RMLMapper [7] to materialise an RDF knowledge graph from source data.

An underlying assumption when generating KGs using mappings is that the data are inherently structured. However, a significant portion of data available on the Internet is, at best, semi-structured. As an example, consider online databases that catalogue medicines and information about them [8, 9]. Notably, information such as the name of a drug or its interactions with other drugs is structured, while

Third Workshop on Knowledge Graphs and Neurosymbolic AI (KG-NeSy 2026), in ISWC 2026 Workshops Joint Proceedings, October 25–26, 2026, Bari, Italy

shikhatkarkee@gmail.com (S. Karkee); e.botoeva@kent.ac.uk (E. Botoeva); sam.coombes@nhs.net (S. Coombes); a.k.jordanous@kent.ac.uk (A. Jordanous); r.o.kafali@kent.ac.uk (Ö. Kafali); dalanti@unibz.it (D. Lanti)
0000-0001-5881-0258 (E. Botoeva); 0000-0003-2076-8642 (A. Jordanous); 0000-0001-9296-2087 (Ö. Kafali); 0000-0003-1097-2965 (D. Lanti)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-05

CEUR-WS.org/Vol-4313/kgnesy-paper25.pdf

other information may come as unstructured text, e.g., dosage instructions, such as *"1 g, every 4–6 hours; maximum 4 g per day"* and *"1 drop of 0.5% solution in the affected eye(s) twice daily"*, and pregnancy warnings, such as *"Not recommended during pregnancy due to lack of sufficient human data"*. Assuming a scenario where the goal is to construct a KG from such a database in order to provide answers to pharmaceutical queries, the dosage and safety information would be expected to have a structured representation. Extracting structured information from unstructured data is therefore necessary to answer such queries.

The standard solution when dealing with unstructured text is to employ large language models (LLMs) [10]. In the context of knowledge graph construction from text corpora, LLMs have been used for named entity recognition, relation extraction, as well as for fact generation [11, 12, 13, 14, 15, 16, 17]. However, the main challenge remains factual correctness, since LLMs are statistical models prone to *hallucinations* [18]—generating plausible-sounding text rather than verified facts. A pure LLM-based solution to knowledge graph construction, therefore, may not be acceptable in a high stakes application, where either the generated KG would be expected to be factually correct, or the correctness of query answers would need to be easily verifiable.

To combine the best of two worlds, the ability of LLMs to intelligently process arbitrary natural language and the correctness guarantees associated with the OBDA approach, we propose leveraging LLMs in a more surgical fashion. Namely our methodology involves integrating LLM queries within the RML mappings, where the input to the LLM is computed by the source part of a mapping rule and the output of the LLM is used to generate RDF terms. Our approach offers several advantages over an end-to-end LLM solution, where an LLM answers user queries directly over the data sources. First, the LLM is not used to deal with information that is already structured and can be processed using standard RML. Second, the LLM is fed only short and focused snippets of text, thus reducing the risk of confusing it. Finally, it allows for a more straightforward correctness verification, as each fact generated by an LLM could be annotated with the text used to extract it.

We evaluate our approach on a Medicines Information (MI) use-case. We use a small JSON dataset created to mimic the existing online MI datasets and a number of SPARQL queries. Our experiments show that integrating LLMs into RML results in fewer errors in the generated KGs and query answers than the pure LLM-based solution, at a lower computational cost.

2. Setup and Evaluation

We leverage the RML-FNML module[1], which permits the specification of arbitrarily complex *user-defined functions* within mapping assertions. This integration enables the dynamic extraction of structured facts from unstructured fields at mapping time, while preserving the declarative nature of RML mappings. We present our idea by instantiating it on a medicines information use-case.

All our artefacts and code can be found at https://github.com/pharmaKG/rmlai.

Data. We consider semi-structured drug information of the kind found in databases such as BNF [8] and EMC [9]. Since their licensing terms restrict the reproduction and redistribution of their content, we created synthetic JSON files for 20 drugs[2]. These contain information on usage (patient group, indications, dosage, and frequency), side effects, and advice for pregnant, breastfeeding, or hepatically/renally impaired patients. We also created separate JSON files with information on how 12 of the above drugs interact with other medicines.

Ontology. We built a simple ontology to capture the data in our dataset along with a SHACL file that defines the schema of a valid RDF graph. There are 5 classes and 24 properties. The values for 13 of these properties need to be extracted from unstructured text — *LLM-extracted* properties. All IRIs and the rest of the property values can be extracted directly from JSON — *immediately accessible* properties.

[1] https://kg-construct.github.io/rml-fnml/ontology/documentation/index-en.html

[2] The original DL 2025 paper reports the experiment results for a different dataset. Here we report the numbers for the dataset published in the github repository, hence the numbers differ slightly

	DeepSeek-7b	DeepSeek-14b	DeepSeek-32b
RML-LLM-KG (198 LLM calls)	79 / 0.05 / 13min	36 / 0.02 / 20min	34 / 0.02 / 40min
LLM-KG (32 LLM calls)	– / – / 23min	742 / 0.53 / 20min	1066 / 0.77 / 40min

Table 1
Error number, ratio of erroneous triples in the generated RDF graph, and the generation time, per LLM (1 run).

Mappings We created RML mapping rules for all our classes and properties. For each class and immediately accessible property, we created standard rules. For each LLM-extracted property, the mapping rule involves a call to a user-defined function implementing an LLM query.

Below we show an extract of mapping rules for the `Dosage` class, the `route` immediately accesssible property and the `amount` LLM-extracted property (instantiated for apixaban):

```
source json_to_table("$.therapeuticPlan[*].dosages[*]", apixaban.json)
target http://drug.uk/dosage/{id} rdf:type drug:Dosage ;
                                  drug:route {routeOfAdministration};
                                  drug:amount grel:getFloatFromLLM( grel:concat(DOSAGE_PROMPT, {doseInfo}) ) .
```

The `source` declaration in the first line essentially creates a relation, where each tuple corresponds to an element of the `dosages` array. In the `target` part, the first two rules are standard RML rules. The rule for the `amount` property states that the object value is computed as `grel:getFloatFromLLM`(p), where *(i)* p is the concatenation of the fixed string `DOSAGE_PROMPT` with the value of `doseInfo`; *(ii)* `grel:getFloatFromLLM` is a user-defined function that prompts an LLM with p and extracts a floating point number from its answer; and *(iii)* `DOSAGE_PROMPT` is the following template:

```
task: Extract dosage amount from the provided text.
output: a json string of format: {"response": DOSAGE_AMOUNT}. DO NOT RETURN ADDITIONAL EXPLANATIONS OR TEXT OR
      MARKDOWN FORMATTING. Only JSON as string.
rule: DOSAGE_AMOUNT is a float value. And it should be a valid xsd:decimal. If no relevant information is
     found in the text, DOSAGE_AMOUNT is null. Do not put null within double quotes.
example: if the text says 50mg daily, DOSAGE_AMOUNT is 50. if it says 50mg in 2 divided dose, as 50 divided by
      2 is 25, DOSAGE_AMOUNT is 25. if text says apply 1 millimeter, DOSAGE_AMOUNT is 1.
text:
```

We evaluate our approach in terms of correctness of the generated knowledge graph and of the answers to SPARQL queries over this graph. For our core experiments, we employed three versions of DeepSeek-R1-Distill-Qwen, an open-source LLM with reasoning capabilities [19], with 7, 14 and 32 billion parameters. We chose these models because they could be run locally, and their increasing sizes let us to explore the trade-off between computational cost and performance. All experiments were run on Intel Xeon Gold 5317 CPU with 32 cores and NVIDIA A100 GPU with 80GB memory.

2.1. Correctness of the Generated Knowledge Graph

We compare the KGs produced by our method using the three DeepSeek LLMs (RML-LLM-KGs) with those obtained by prompting the same models to convert the JSON files directly into RDF (LLM-KGs). We evaluate both against a manually constructed ground-truth KG. For all models, we set the temperature parameter to 0 achieving deterministic behaviour (confirmed empirically), thus requiring a single run.

Across 20 drugs, the total number of triples in the extracted RDF graph was 1,384. Out of these, the object value of 550 triples was LLM-extracted. For each KG, we summarise the number and ratio of erroneous triples in the generated RDF graph, and report generation time in Table 1.

For KGs generated using LLM-enhanced RML mappings, the smallest model performed worst (0.14 error rate for LLM-extracted values and 0.05 error rate for the whole graph), and one run took around 13 minutes. The two bigger models performed similarly (0.06 error rate for LLM-extracted values and 0.02 error rate for the whole graph), while DeepSeek-14b was about twice faster than DeepSeek-32b.

For KG generation using LLMs only, we asked them to convert the JSON data into an RDF graph in N-Triples format, processing each JSON file independently. This avoids combining all JSON files into one big context, potentially reducing the number of errors. The prompts included a verbalised version of the ontology (a plain English summary, including possible values for properties), the naming conventions

	DeepSeek-7b	DeepSeek-14b	DeepSeek-32b
NL-over-LLM	24 / 110s	22 / 180s	19 / 300s
NL-over-JSON	17 / 90s	8 / 150s	6 / 330s
SPARQL-over-RML-LLM-KG	3	2	2

Table 2
Number of incorrect query answers out of 28, per query answering approach and LLM. For the natural language queries, we report the average time it took to answer all 28 queries with LLMs.

for URIs for easy comparison with the ground truth, and the same instructions for LLM-extracted properties as in the prompts used in the in the mappings.

The smallest model failed to produce a usable RDF graph, with syntactic and semantic errors beyond repair. We manually fixed some minor errors (such as such as adding prefix declarations, adding quotes to literals, redundant spaces in IRIs) in the outputs of the other two models to allow for some non-trivial evaluation results. In the corrected KGs, there were about 50% of incorrect triples for the 14b model, and 77% for the 32b model. The observed errors included using non-existent properties, hallucinated prefixes in IRIs, missing triples.

We consider our approach successful at generating "almost correct" knowledge graphs from semi-structured information: 34 errors across 1,384 facts yield an error rate of just 2.5%. By contrast, a pure LLM-based solution produced substantially more errors even on small JSON files, a gap that would likely widen on more realistic, larger inputs. Another advantage of our solution is that it guarantees the conformance of the generated graph with the ontology. A pure LLM-based solution offers no such guarantee: it may produce triples with incorrect vocabulary.

2.2. Answers to queries

We evaluate our approach through SPARQL query answering over the generated RDF graphs and compare it with *(i)* questions answered directly from the LLM's internal knowledge (NL-over-LLM); and *(ii)* questions answered from individual JSON drug documents in a RAG-like setting (NL-over-JSON).

We selected 28 queries, loosely based on actual MI enquiries, to be answered over our dataset. Each query is over a specific drug. The first 8 queries are expected to return one of three values *Yes*, *No* or *Unknown*. The latter value should be returned when there is no evidence to back up a positive or a negative answer. The remaining 20 queries ask the dosage amount and unit for an indication, for each of the 20 drugs in our dataset. The correct answers were determined through the ground truth graph.

We employed zero-shot prompting strategy for the LLM queries in NL-over-LLM and NL-over-JSON. Based on our initial experiments, we asked LLMs to provide an answer in a structured format. Below are the zero-shot instructions we used for the dosage queries:

```
Rule:
- Provide the answer in a structured plain text JSON format
- JSON should contain "dosage amount" and "dosage unit" as keys and their respective values.
- Do not include any additional information or explanation.
- Do not provide JSON in markdown formatting. Only provide plain text JSON.
```

We summarise the numbers of incorrect query answers in Table 2. Asking for answers directly from LLMs results in over 19 errors for all three models. Providing LLMs relevant information in a JSON file reduces the number of errors to 17 for the smallest, 8 for the medium and 6 for the largest models. The number of errors in SPARQL query answers over the three generated graphs ranges from 3 for the smallest LLM, to 2 for the medium and the largest ones.

The comparison shows that our approach advances the existing work along two dimensions. First, with respect to the standard RML mappings, our LLM-enhanced mappings make more data accessible for structured query answering, whose benefits include better explainability of the answers. Second, providing LLMs with fine-grained context in LLM-enhanced mappings can reduce the risk of incorrect answers and the computational cost, compared to probing the implicit knowledge of an LLM as in NL-over-LLM, or to providing a potentially much bigger context as in NL-over-JSON.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati, Linking data to ontologies, J. on Data Semantics (2008) 133–173. doi:10.1007/978-3-540-77688-8_5.

[2] S. Harris, A. Seaborne, SPARQL 1.1 Query Language, W3C Recommendation, W3C, 2013. Available at http://www.w3.org/TR/sparql11-query.

[3] R. Cyganiak, D. Wood, M. Lanthaler, RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation, W3C, 2014. Available at http://www.w3.org/TR/rdf11-concepts/.

[4] S. Das, S. Sundara, R. Cyganiak, R2RML: RDB to RDF Mapping Language, W3C Recommendation, W3C, 2012. Available at http://www.w3.org/TR/r2rml/.

[5] A. Dimou, M. Vander Sande, J. Slepicka, P. Szekely, E. Mannens, C. Knoblock, R. Van de Walle, Mapping hierarchical sources into RDF using the RML mapping language, in: Proc. of ICSC, IEEE, 2014, pp. 151–158. URL: https://doi.org/10.1109/ICSC.2014.25. doi:10.1109/ICSC.2014.25.

[6] A. Iglesias-Molina, D. V. Assche, J. Arenas-Guerrero, B. D. Meester, C. Debruyne, S. Jozashoori, P. Maria, F. Michel, D. Chaves-Fraga, A. Dimou, Rml ontology portal: Modular resources for the rdf mapping language, https://kg-construct.github.io/rml-resources/portal/, 2023. Accessed: 2025-06-05.

[7] RML.io, RMLMapper: A java implementation for executing RML rules to generate Linked Data, https://github.com/RMLio/rmlmapper-java, 2025. Version 7.3.3, MIT License.

[8] Joint Formulary Committee, British national formulary, https://bnf.nice.org.uk/, Accessed: 2025-05-15.

[9] Electronic medicines compendium, https://www.medicines.org.uk/emc/, Accessed: 2025-05-15.

[10] B. Min, H. Ross, E. Sulem, A. P. B. Veyseh, T. H. Nguyen, O. Sainz, E. Agirre, I. Heintz, D. Roth, Recent advances in natural language processing via large pre-trained language models: A survey, ACM Computing Surveys 56 (2023) 1–40.

[11] K. Hakala, S. Pyysalo, Biomedical named entity recognition with multilingual BERT, in: Proc. of the 5th Workshop on BioNLP Open Shared Tasks, Association for Computational Linguistics, 2019, pp. 56–61. URL: https://aclanthology.org/D19-5709/. doi:10.18653/v1/D19-5709.

[12] C. Alt, M. Hübner, L. Hennig, Improving relation extraction by pre-trained language representations, arXiv preprint arXiv:1906.03088 (2019).

[13] L. Yao, C. Mao, Y. Luo, Kg-bert: Bert for knowledge graph completion, arXiv preprint arXiv:1909.03193 (2019).

[14] I. Melnyk, P. Dognin, P. Das, Grapher: Multi-stage knowledge graph construction using pretrained language models, in: Workshop on Deep Generative Models and Downstream Applications, 2021.

[15] T. Ayoola, S. Tyagi, J. Fisher, C. Christodoulopoulos, A. Pierleoni, ReFinED: An efficient zero-shot-capable approach to end-to-end entity linking, in: Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, 2022, pp. 209–220. doi:10.18653/v1/2022.naacl-industry.24.

[16] G. Ciatto, A. Agiollo, M. Magnini, A. Omicini, Large language models as oracles for instantiating ontologies with domain-specific knowledge, Knowledge-Based Systems 310 (2025) 112940. URL: https://www.sciencedirect.com/science/article/pii/S0950705124015740. doi:https://doi.org/10.1016/j.knosys.2024.112940.

[17] B. Mo, K. Yu, J. Kazdan, P. Mpala, L. Yu, C. I. Kanatsoulis, S. Koyejo, KGGen: Extracting knowledge graphs from plain text with language models, in: The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2025.

[18] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, et al.,

A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions, ACM Transactions on Information Systems 43 (2025) 1–55.
[19] DeepSeek-AI, Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL: https://arxiv.org/abs/2501.12948.