Automatically Building and Updating a Knowledge Graph of MLIP Models

Alexis Beer[1,†], Liudmyla Klochko[1,2,*,†] and Mathieu d'Aquin[1,2,†]

[1]MosAIk Team, LORIA, Université de Lorraine, CNRS, Nancy, France
[2]CRET AIREL, Université de Lorraine, Nancy, France

Abstract

Complementing the many efforts in providing semantic representations of concepts, notions, and entities in materials science, we report and illustrate a process by which we can automatically build a knowledge graph of the fast evolving field of machine learning applied to the prediction of material properties, focusing on MLIP (Machine Learning Interatomic Potential). This LLM-based process relies on multiple steps, from information extraction in documents and articles to a validation loop using SHACL constraints to detect and correct errors. It is carried out on a model-by-model basis, focusing on the consistency of representation, therefore enabling an iterative construction where the addition of new models is facilitated. We illustrate the process by showing a few interesting aspects that can be queried from a knowledge graph built from the models listed in the Matbench Discovery leaderboard.

Keywords

MLIP, machine learning model, materials science, knowledge graph, ontology

1. Introduction

The application of AI methods, especially deep learning-based models, to materials science is progressing very quickly. In just the last couple of years, we have seen such applications evolve from simple prediction models of specific physical properties to the use of Physics-Informed Neural Networks (PINNS) [1] and the development of specific types of GNNs. One of the fastest areas of development, at the time of writing, concerns foundational machine learning interatomic potential (MLIP) [2] or force field (MLFF) models that are trained on millions of numerical simulations, mostly based on atomic structures, and can be used/adapted/fine-tuned to enable many applications. This can be seen particularly through the Matbench Discovery leaderboard[1], where the top ranked models are GNN-based MLIP models, with new competitive models being introduced on a regular basis.

Those models have the potential to fundamentally change the way materials science operates. However, as the field evolves so quickly and new models appear constantly, it also becomes harder to keep track of those developments and exploit them effectively. What new models have been proposed? What distinguishes them from already existing ones? How have they been trained? Are they suitable for the specific materials/conditions relevant to a particular application? Answering these questions requires continuous monitoring of benchmark results, leaderboards, and the rapidly growing scientific literature.

In this paper, we propose summarizing all the relevant information about those models into a constantly evolving knowledge graph to effectively support this monitoring. Such a process requires guidance from an ontology that is tailored for the representation of deep learning-based models suitable for describing MLIP models, their training, and evaluation. Our goal is to obtain a detailed description of those aspects of MLIP models, with information that is currently scattered across unstructured sources.

SeMatS 2026: The 3rd International Workshop on Semantic Materials Science co-located with the 25th International Semantic Web Conference (ISWC 2026), October 25th, 2026, Bari, Italy

*Corresponding author.
†These authors contributed equally.
alexis.beer@loria.fr (A. Beer); liudmyla.klochko@loria.fr (L. Klochko); mathieu.daquin@loria.fr (M. d'Aquin)
0000-0003-2468-8738 (L. Klochko); 0000-0001-7276-4702 (M. d'Aquin)

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

[1]https://matbench-discovery.materialsproject.org/

CEUR-WS.org/Vol-4299/paper4.pdf

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-10-21

Indeed, considering the rapid pace of development of such models, we cannot assume that a manual population of the knowledge graph would be sufficient. For this reason, we propose a Large Language Model (LLM)-based information extraction and knowledge graph construction process. This process is guided by a specifically created ontology to obtain information that instantiates it with data obtained from articles and other textual resources[2] that describe each model. One particularity of the process we propose is that we not only include validation of the resulting knowledge graph through a suite of tests implemented by SHACL (Shapes Constraint Language) constraints [3], but we also include an LLM-based approach to correct errors detected by this process.

This paper is structured as follows: first, we summarize the elements of related works that have been useful in the development of our knowledge graph construction process. Then, we provide an overview of the approach and a section for each of the main components of the process: the necessary ontology to be used by the process, the LLM-based information extraction method, and the LLM- and SHACL-based error correction method. Finally, we conclude on the state of the created knowledge graph for the models listed under the Matbench Discovery leaderboard. All the code, ontologies and results for this process are available in a public repository[3].

2. Related work

Materials science is a highly interdisciplinary field that combines aspects of materials at different scales and from different points of view: theoretical, experimental, application, etc. As a result, many works have focused on providing ways to semantically describe and structure various aspects of the field, including studies that relied on applying ontologies and knowledge graphs to materials science [4, 5]. The goal of this section is not to provide a complete account of those works. Generally, they tend to be motivated by similar arguments to those that motivate the construction of ontologies and knowledge graphs in other sciences [6]. In particular, knowledge graphs and ontologies support semantic interoperability, and materials science, like many other sciences, is based on information that varies in form and nature, including measurements, experiments, and computations described in databases, articles, and other repositories. Many applications of knowledge graphs and ontologies in the field have therefore been related to the integration of heterogeneous data sources [7, 8]. Closer to the work in this paper, a number of studies have started looking into automating the extraction of information about materials (and related concepts) from articles and other textual resources that could be used to build knowledge graphs and ontologies. These include works to create embeddings of materials from the literature, such as the mat2vec model [9]. They also include specific NLP pipelines adapted to the extraction of entities related to materials [10]. More recently, automatic processes for extracting entire knowledge graphs [11] have also started appearing, including those relying on LLMs [12, 13]. In this work, we focus on building a knowledge graph of machine learning models for materials, especially focusing on MLIP models, which, to the best of our knowledge, has not yet been explored.

Since our focus is on machine learning models for materials science, related works also include ontologies and knowledge graphs that are dedicated to the representation of machine learning models. Surprisingly, relatively few works have been dedicated to this topic. The ML Schema [14] is one example, promoted by a dedicated W3C community group, that provides a set of classes and properties for describing machine learning experiments. It relies on other ontologies to describe models, their implementations, and evaluations at a high level. Similarly, SML (Semantic Machine Learning Model Ontology [15]) describes machine learning models, the datasets on which they rely, and notions of evaluation. It is more focused on usability for describing practical applications rather than experiments. Finally, FAIR4ML[4] is an ontology-based extension of Schema.org specifically targeting the FAIRness of

[2] in MatBench Discovery, models are described through pdf articles, a YAML file that provides some basic information, and which also links to datasets relevant to the evaluation of models. This is what we mean by articles and textual resources.

[3] https://github.com/alexisbe-github/Matbench-onto

[4] https://github.com/RDA-FAIR4ML/FAIR4ML-schema

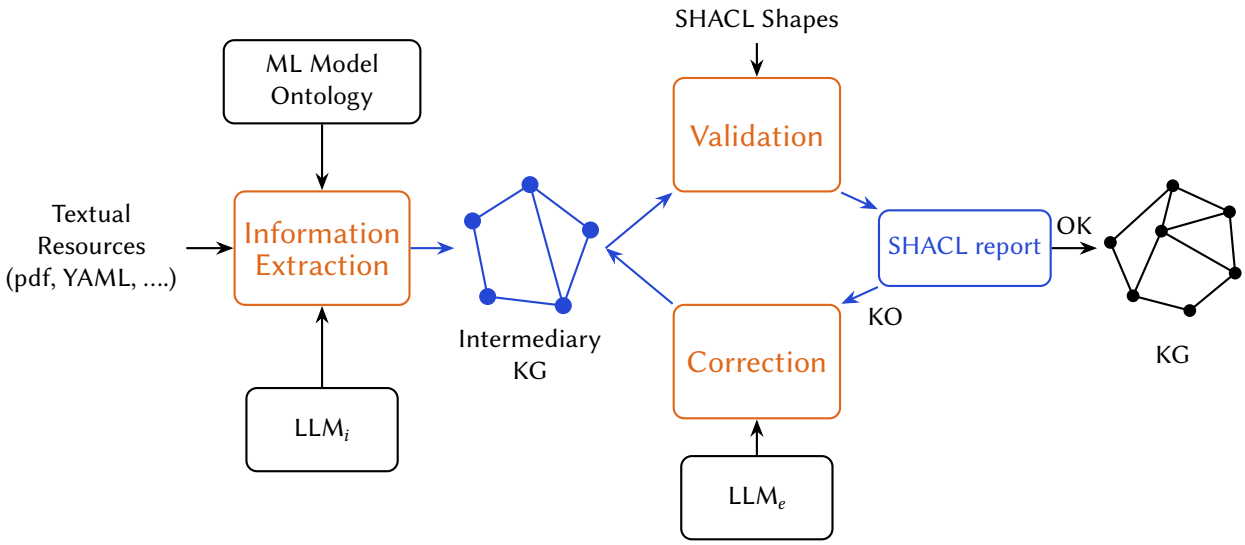


Figure 1: Overview of the automated knowledge graph construction process.

ML model documentation, with classes and properties for evaluation metrics, intended applications, and tasks. There are a few other ontologies, vocabularies, and thesauri that we could have cited here. As described later, our objective is to describe the fast evolving MLIP models used to predict the physical properties of materials in a granular way, so that those descriptions could be used, for example, for model recommendation. This implies representing their architectures, training parameters, versions, variants, evaluations, and specific prediction results. Since we focus here on building a proof-of-concept process for information extraction that is guided by an ontology, we rely in this first step on a specifically made ontology and do not describe the methodology for creating it in details. We aim to refine the ontology itself, and align it with existing ones in materials science and on machine learning models, as part of our future work.

3. Overview of the approach

Figure 1 provides an overview of the process presented in this paper. The process starts with an information extraction step that takes as input textual resources describing a particular model. Here, pdf and YAML are mentioned, as those are the main formats for describing the models participating in the leaderboard on the Matbench Discovery portal. Information extraction here is carried out through the use of a LLM (see Section 5) and is guided by our ML Model Ontology (see Section 4) in the sense that it is aimed not only at producing a knowledge graph that is structured according to the ontology, but it also relies on the ontology as a resource to identify information to extract (ask the right questions). A key innovation in our process is the validation-correction loop (see Section 6). This process takes SHACL Shapes [3] that act as integrity constraints or unit tests for the created knowledge graph of a given model. The knowledge graph is therefore validated against those constraints. If any inconsistency is detected, another LLM is used, taking as input the original resources, the created knowledge graph, and the SHACL validation report, and is instructed to find ways to rectify the inconsistency. This process is carried out in a loop until all inconsistencies have been removed, ending the process with a cleanly structured knowledge graph for the considered model. That the knowledge graph for a model is defined according to our ontology and validates our SHACL constraints ensures that it can be integrated into a global knowledge graph of all models. In this way, the global knowledge graph can be easily updated when a new model appears. However, the absence of SHACL violations only demonstrates conformance with the explicitly defined constraints and does not guaranty the full completeness or factual correctness of the resulting knowledge graph.

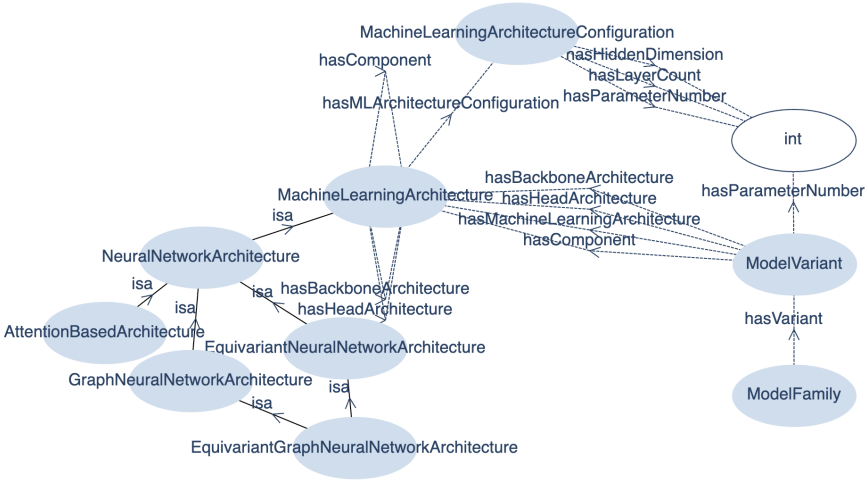


Figure 2: Diagram of an extract from the core (architecture) ML Model Ontology. (`isa` means `rdfs:subClassOf`)

4. An Ontology of ML Models

As mentioned before, a fine-grained ontology is required for the description of machine learning models to be used for the knowledge graph of MLIP models. Creating a formally validated, aligned, and reusable ontology for describing MLIP models is an entire work in itself, which we choose to leave for future work. The core aspects of the methodology to build it include the requirements for it to describe models, including all the ways in which they are present on Matbench Discovery, detailing their architecture, the data on which they were trained, and their evaluation. A key consideration was also to take into account that MLIP models often evolve and may be grouped into model families, variants, etc. Finally, while we include evaluation results, with metrics, tasks, and benchmarks to be usable for model recommendations, the specific predictions and expected values on individual test inputs must also be included.

Following these requirements, the ontology is modularized into three main components: the core that defines models, their relations, and architectures; the training module that describes training strategies, hyperparameters, and data; and the evaluation module, which describes both global metrics run on benchmarks and results from individual inputs.

4.1. The core ML Model Ontology

Figure 2 visually summarizes the core ML model ontology. As can be seen, the main class is called ModelVariant because it is rare for what we would usually refer to as an MLIP model to exist in only one version. It is however important that when we describe a model, we identify the exact version we are describing. For the more general notion of a model, the one for which there can be multiple variants, we use the class ModelFamily. As can be seen, besides this aspect, the core ontology focuses mostly on describing models (variants) through their architecture. The architecture class itself enables the description of architectures recursively, through components, and specific properties are defined for components that take a particular role (backbone, head, etc.) In this way, we can, for example, describe a GNN model with an MLP-based classification head. The architecture can also be associated with a configuration that enables specifying basic descriptors for the architecture (or its components), such as the number of parameters or the number of layers.

4.2. The ML Training Module

The training module, as summarized in Figure 3, mostly focuses on describing the process by which a model variant has been produced. The central class is therefore MachineLearningTrainingRun, which connects to the ModelVariant class of the core ontology. It also includes a number of classes and

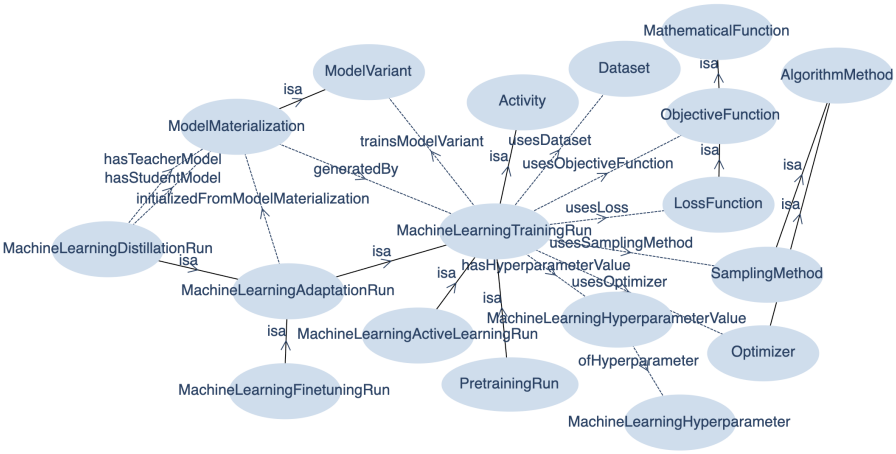


Figure 3: Diagram of an extract from the training module. (`isa` means `rdfs:subClassOf`)

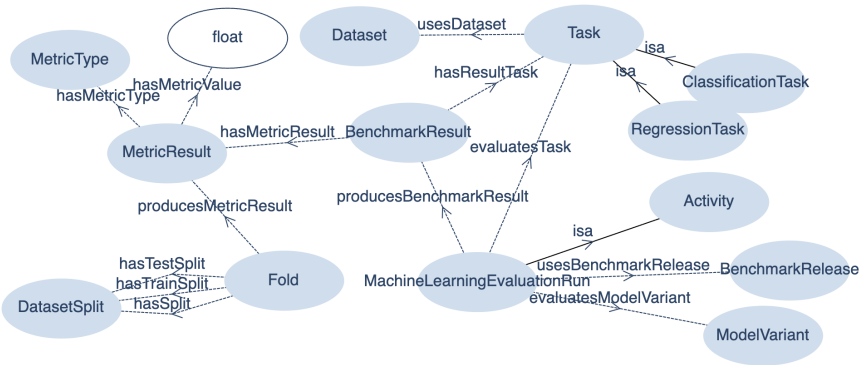


Figure 4: Diagram of an extract from the evaluation module. (`isa` means `rdfs:subClassOf`)

properties to describe the setup of the training run, with aspects such as the dataset, the loss function, the optimizers, and the hyperparameters used. A large part of the module is also dedicated to describing different training strategies, some of which, such as distillation or fine-tuning, take existing models (materializations, which are concrete representations/implementations of model variants) as input.

4.3. The ML Evaluation Module

The evaluation module is summarized in Figure 4 and includes classes for the tasks being carried out and the metrics used to measure the performance of models. The central class is the MachineLearningEvaluationRun, which represents the evaluation of a ModelVariant on a particular task. Each evaluation run produces a BenchmarkResult, which groups the resulting MetricResult instances.

One aspect that is not represented in the figure is the evaluation of results on individual inputs. This aspect is relatively simple, as it involves attaching a prediction result to a model variant that provides an identification of the input, as well as the predicted value and the expected value. Based on this, evaluation metrics on subsets of the datasets can be easily re-calculated.

5. LLM-based Process for Extracting Model Details from Documents

As already described in Section 3, the information extraction process is designed to extract the part of the knowledge graph around one particular model (or rather, one model variant). The starting point is therefore textual documents that are supposed to collectively contain all the details needed to fully

describe a model according to our ontology (including its training and evaluation). In the rest of the paper, the process will be illustrated using resources available through the Matbench Discovery portal. For each model, a pdf file serves as the reference article in which the authors have described the model, and a YAML file contains the Matbench description of the model, including the results of evaluations. The results of individual predictions are treated separately, as they can be interpreted through a simple script without requiring the use of a LLM.

In order to guide the LLM in identifying the relevant pieces of information, a JSON file is created that maps each property of relevant objects around the model variants to a question in English for the LLM to be queried with. In order to facilitate the process, taking into consideration that the syntax of RDF (Resource Description Framework[5]), which we ultimately want to use for our knowledge graph, is not as familiar as that of JSON, the process is divided into two steps:

1. **Information extraction.** The LLM is first asked to extract all the information corresponding to all those questions to produce one JSON file reflecting this structure and including the extracted information values.
2. **Knowledge representation.** In a separate process, the extracted JSON file is converted into RDF using the ontology as a semantic reference. The LLM is provided with both the JSON content and the relevant ontology classes and properties and is asked to generate RDF statements that conform to the ontology.

One interesting aspect of the process described here is that it not only obtains the extracted information (the values of properties directly or indirectly attached to the model variant individual in the knowledge graph). The LLM is also prompted to provide evidence for every extracted piece of information. Such evidence takes the form of a reference to the paragraph or section of the document in which the information can be found, or a quote from the document. For example, for the equiformer-v3-mp model[6], the initial JSON-based extraction might contain the following information regarding the number of layers:

```
" architecture": {
   ...
   "layer_count": {
        "value": 7,
        "evidence": "PDF Appendix Table 10: Number of Transformer blocks = 7
                     for MPtrj training"
    },
    ...
```

clearly indicating that the architecture of the model contains 7 layers, representing 7 blocks of transformers, and that this information is available in Table 10 of the appendix of the provided pdf document.

Those pieces of evidence represent crucial information regarding the provenance of various elements of the knowledge graph, which can be used to verify and validate the values obtained for various properties and to identify their sources. We therefore integrate it directly into the RDF representation of the information through the use of RDF-star[7]. Indeed, RDF-star enables the integration of triples about triples in RDF. In this case, the information about the number of layers of the model will be represented as a triple between its architecture's configuration and the number 7. The source of this information can therefore be added as an annotation of this triple in the following way in RDF-star:

```
<< mlip:equiformer_v3_mp_equiformerv3_architecture_configuration
                   mlo:hasLayerCount "7"^^xsd:integer >>
                           prov:wasDerivedFrom "PDF Appendix Table 10:
```

[5]https://www.w3.org/RDF/
[6]https://huggingface.co/mirror-physics/equiformer_v3
[7]https://w3c-cg.github.io/rdf-star/cg-spec/editors_draft.html

```
Number of Transformer blocks = 7 for MPtrj training" ;
dcterms:source <https://arxiv.org/pdf/2604.09130.pdf> .
```

Hence, the produced knowledge graph can be queried not only for detailed information about each model but also for the precise origin or provenance of the information.

6. SHACL-based Validation and Correction of the Extracted Knowledge Graph

We cannot naturally assume, even with large, recent LLMs, that the information extracted will always be correct and adequately represented. Including the source of each piece of information in every triple, as described above, can help with spot-checking a subset of the generated triples for accuracy. However, carrying out such verifications for every aspect of the knowledge graph would be counterproductive to our objective of automating the construction of the knowledge graph. For this reason, we also include automatic validation of the produced knowledge graph for each model, which enables us to set up another LLM-based process for error correction.

For this process, we naturally rely on SHACL. Indeed, SHACL enables writing constraints (shapes) that should be verified by the knowledge graph and ensures that they indeed are. Based on our design of the ontology as well as the recommendations from SoCK [16], we built a number of SHACL shapes to act as unit tests, completeness tests, and integrity constraints for the constructed knowledge graph. For a given model, the created knowledge graph will therefore be validated, leading to a SHACL report listing all the violated constraints. Using this report, which indicates errors in the issue of representation and the original sources, a second LLM (which does not have to be the same as the first) is therefore instructed to correct the identified issues. This process is repeated in a loop until no new issues are identified by SHACL, leading to a knowledge graph that is valid according to the established constraints. For example, in the initially generated knowledge graph for AlchemBERT[8], the individual `alchembert_pretraining` did not specify the dataset used during pre-training. This violated a `sh:minCount 1` constraint requiring every `train:PretrainingRun` to have at least one `train:usesDataset` relation. Based on the SHACL report and the original sources, the repair LLM added the following missing triple:

```
trind:alchembert_pretraining train:usesDataset dataind:mp_2022 .
```

The subsequent SHACL validation confirmed that this violation had been resolved.

7. Results and Discussion

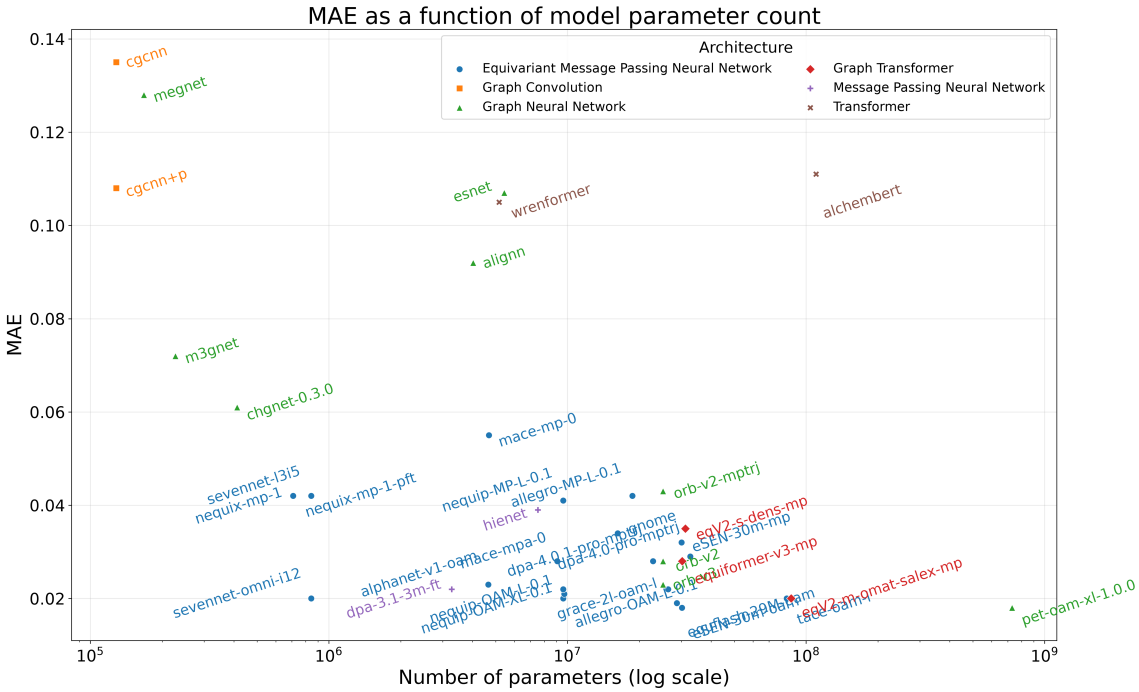
We applied the process described above to all the models of the Matbench Discovery portal using Meituan's LongCat-2.0-Preview LLM[9] for both information extraction (LLM_i in Figure 1) and error correction (LLM_e in Figure 1).[10] This resulted in a knowledge graph containing 49 model variants from 31 model families.

Having such a knowledge graph enables us to query and explore the landscape of MLIP models in a more structured way. For example, SPARQL queries can be used to identify the most frequently used training datasets, compare model architectures, and see the relationship between model size and its predictive performance. This is what is shown, for example, in Figure 5(a), where one of the performance metrics used in Matbench Discovery is plotted against the size of models in terms of the number of parameters. Unsurprisingly, we can see a general trend by which larger models tend to obtain better performance, with pet-oam-xl-1.0.0 being the largest and obtaining the lowest MAE, and

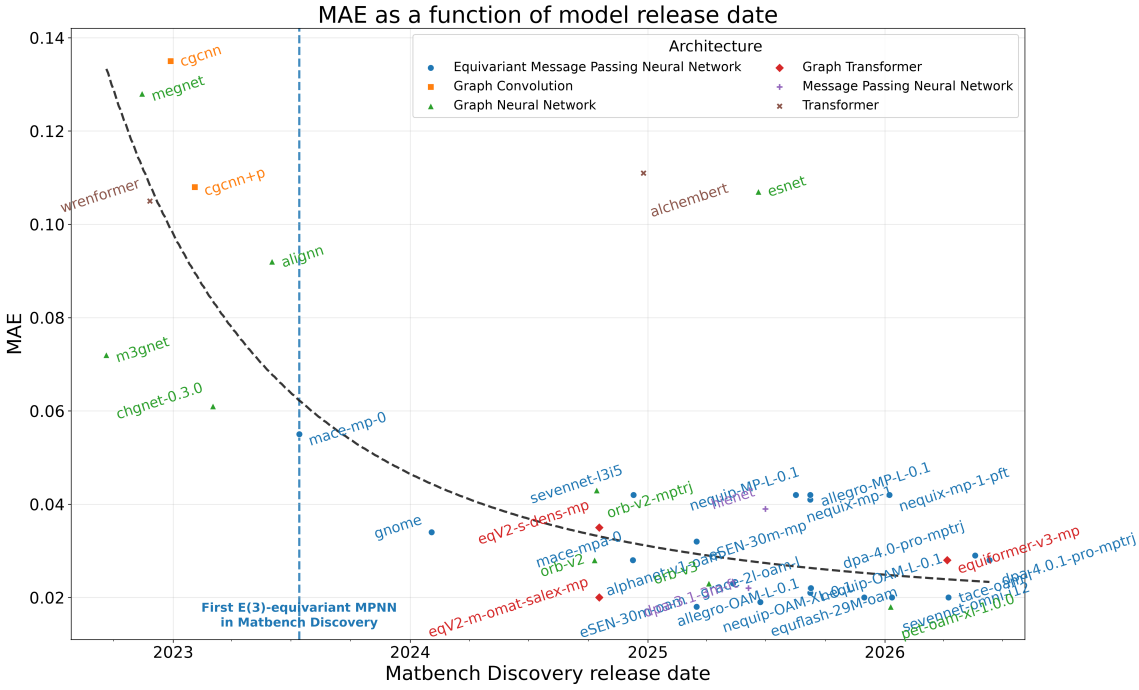
[8] https://gitee.com/liuxiaotong15/alchemBERT

[9] https://www.longcatai.org/models/longcat-2

[10] The reason for using this LLM was that it was free in stealth mode on OpenRouter under the name OWL Alpha at the time and achieved very high performance.



(a) Model performance as a function of the number of parameters. Colors indicate the architecture type.



(b) Model performance as a function of the Matbench Discovery release date. Colors indicate the architecture type.

Figure 5: Relationship between model performance, model size, and release date. MAE here refers to the mean absolute error on the convex hull distance regression task. Lower MAE values indicate better performance.

cgcnn being among the smallest and obtaining the highest MAE on this task. Clear outliers with respect to this trend can, however, be observed, such as alchembert being a relatively large model, obtaining relatively poor performance.

Figure 5(b) is similar to Figure 5(a), plotting performance according to the same criteria but against

the date at which a model was added to the Matbench Discovery leaderboard. Here again, the downward trend (with lower and lower differences) is not surprising, as it indicates that newer models tend to perform better than older ones. There is also a clear correlation between the year of release of a model and its size (models are becoming bigger with time). While we can again notice significant outliers to the trend here in time, it is, however, more pronounced than the one above, showing that recency tends to be a better indicator of performance than size.

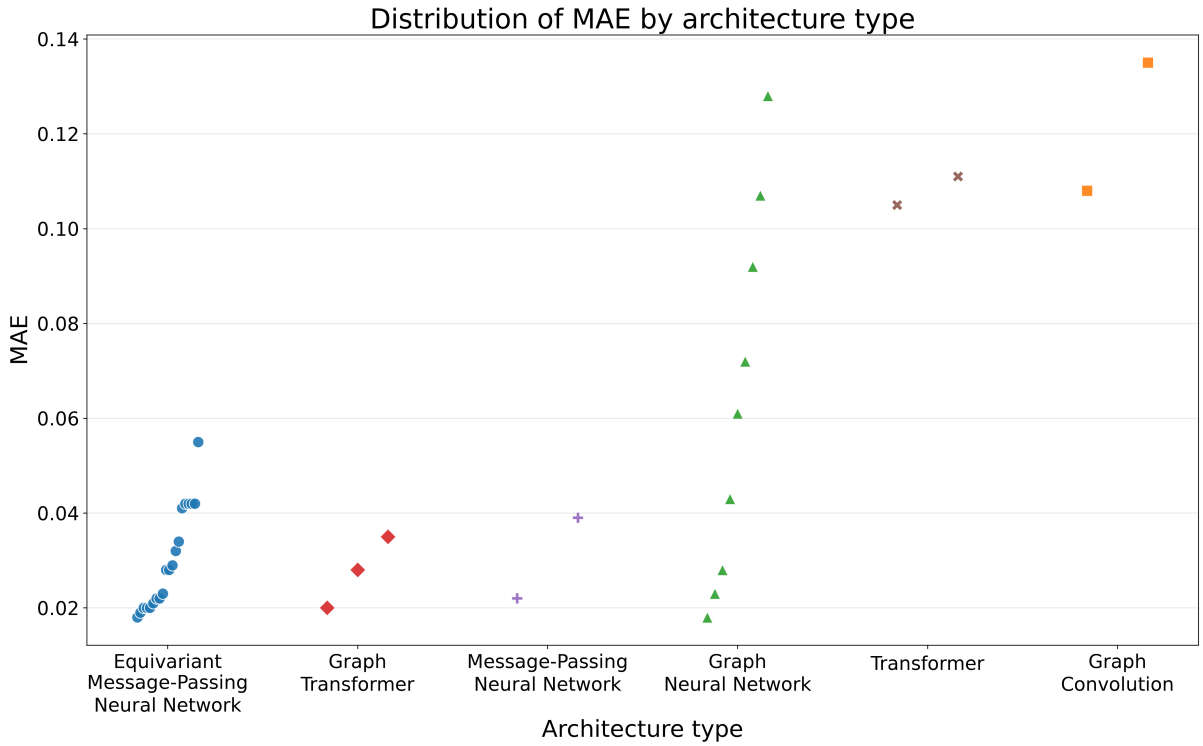


Figure 6: Model performance (MAE) across architecture types on the convex hull distance regression task. Lower values indicate better predictive performance.

Both previous subfigures of Figure 5 also show the architecture type used by each model (through colors). This provides interesting information; for example, as marked on Figure 5(b), regarding the time at which specific types of architecture appeared (here E(3)-equivariant MPNN) and how they evolved over time. We can also see, for example, how early attempts at graph convolution were not pursued, while, in contrast, general graph neural network approaches remained in use and continued to improve (with exceptions). We can also notice the late appearance of graph transformer approaches, with models placing themselves among the best of their time. An interesting anecdotal point is that it appears from both figures that the two attempts at using transformer architectures did not seem to be particularly successful on this specific task, especially considering their size. While we provide these visualizations and discussions to illustrate the value of the knowledge graph, it shows that it could be interesting to further explore these aspects, looking at other prediction tasks and specific sets of materials.

The relationship between model architecture and performance is depicted more precisely in Figure 6, showing the distribution of MAE for every architecture type. This somehow confirms what was already seen previously, including that the few cases of applications of graph convolution, while effective at the time, were not considered sufficiently promising and were not pursued further. Similarly, pure transformer based architectures did not, at this point, achieve competitive performances. On the other hand, more recent innovations such as equivariant message passing architectures and graph transformers perform consistently well on the considered task, even though it can be noted that more general architectures such as message passing neural networks and general GNNs can also achieve

good performance. In our process, the GNN architecture was attributed to any model that relied on a graph neural network that could not be attributed to a more specific one (such as equivariant message passing neural networks or graph transformers). This explains why, in terms of performance, this architecture ranges from among the worst to among the best, as it can include models that are built in significantly different ways. Having said that, all the graph-based methods, except for graph convolution, have models that have achieved competitive performances, with MAEs at or below 0.02, showing that other aspects (such as the size or the training dataset used) might be more important than the architecture.

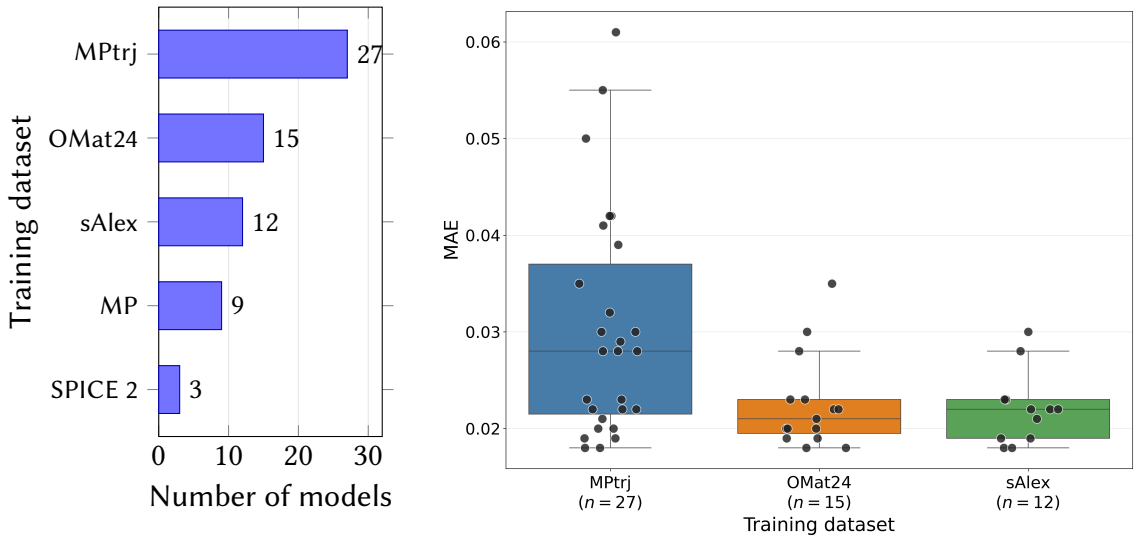


Figure 7: Training-dataset usage and associated predictive performance. Left: the five most frequently used training datasets, measured by the number of distinct model variants. Right: distribution of MAE for models trained on MPtrj, OMat24, and sAlex. Models trained on multiple datasets are included in each corresponding group.

Regarding training datasets, Figure 7 shows information about the most commonly used datasets for pre-training, fine-tuning, or other training processes in the models considered here. It should be noted that a number of models use more than one dataset for training, which explains why the sum of the counts of dataset usage is higher than the number of models included in the knowledge graph. The dataset that is, by far, the most commonly used is MPtrj, which consists of extracts of information, structures, and properties from the Materials Project[11]. OMat24[12] and sAlex[13] come second and third, but it could be easily noticed from the Matbench Discovery leaderboard that those three datasets are often used together. This is actually visible in the figure showing the distribution of performances (again, in terms of MAE, right of Figure 7) according to the dataset used. From this, it can be seen that, on average, models trained with OMat24 and sAlex achieve significantly better performances than those trained on MPtrj alone. However, it can also be seen that many of the models trained on sAlex were also trained on OMat24 and on MPtrj, while a large number of models were trained on MPtrj alone. Rather than suggesting that OMat24 and sAlex are better datasets, this seems to indicate that training on multiple datasets tends to be a more successful approach than focusing only on one.

8. Conclusion

In this paper, we have described a process by which we can automatically and incrementally build a knowledge graph of machine learning models for material property predictions, focusing especially

[11]https://next-gen.materialsproject.org/
[12]https://huggingface.co/datasets/facebook/OMAT24
[13]a subset of Alexandria - https://alexandria.icams.rub.de/

on MLIP models as present in the Matbench Discovery leaderboard. This approach includes several innovative aspects, from the recording of evidence for specific elements of information (triples in the knowledge graphs) corresponding to parts of documents in which the information was found by an LLM, to an LLM and SHACL-based validation/correction loop that ensures that the structure through which models, their training, and their evaluation are described in the knowledge graph is consistent across models. We illustrated this process by creating a knowledge graph of the models present in the Matbench Discovery leaderboard and querying it to build interesting visualizations of various aspects of those models and their performance.

There are many aspects of the proposed process that could still be improved, especially regarding its validation. Indeed, the SHACL-based loop put in place, while effective in guaranteeing an appropriate and consistent structure for the information extracted by LLMs, cannot guaranty that the information in question is factually correct. Including specific evidence for extracted triples in the knowledge graph through RDF-star can be helpful for manual validation, but this remains a complicated and time consuming task. Gold standard-based evaluations here are not feasible since building the part of the knowledge graph describing a model is an excessively complex and time consuming task. Part of our plan is therefore to make such a knowledge graph public, together with tools to support building visualizations, analyzes, and recommendations that will be attached to community feedback mechanisms. Another part is to extend the framework to include LLM-based validation and LLM ensembles to minimize the hallucination rate and the risks of incorrect content. Such methods would also enable us to identify low confidence statements on which to focus (community-based) human validation.

Finally, another avenue for future work actually relates to those tools and, more broadly, the use of such a knowledge graph. Indeed, we illustrated some of the ways in which the knowledge graph could be queried to analyze how some factors might affect the performance of models on a specific task (convex hull distance regression). There are many tasks that could be considered, along with many other factors that could be analyzed for such investigations, which we hope a future public release of the knowledge graph will enable. One aspect that appears particularly interesting is to consider performance in relation to the characteristics of the materials being studied. This is what motivated our inclusion of the specific predictions of each model on materials included in test datasets. Indeed, metrics such as MAE are aggregated over all tested materials, but it can be expected that (in particular, depending on their training set), some models would perform differently on different subsets of materials. Assessing the relationship between materials' properties and the ability of models to accurately predict other properties for those materials seems valuable not only to better understand the models but also to identify the most suitable ones for specific tasks on specific sets of materials. A key next step for the work presented here is, therefore, to build such a recommendation engine for material prediction models.

Declaration of use of Generative AI

Generative AI was used within the process described in this paper (LLM based information extraction and correction), as well as in support of the development activities. Generative AI models were also used for grammar and stylistic corrections/improvements of the text of this paper.

References

[1] M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686–707. URL: http://dx.doi.org/10.1016/j.jcp.2018.10.045. doi:10.1016/j.jcp.2018.10.045.

[2] R. Jacobs, D. Morgan, S. Attarian, J. Meng, C. Shen, Z. Wu, C. Y. Xie, J. H. Yang, N. Artrith, B. Blaiszik, G. Ceder, K. Choudhary, G. Csanyi, E. D. Cubuk, B. Deng, R. Drautz, X. Fu, J. Godwin, V. Honavar, O. Isayev, A. Johansson, B. Kozinsky, S. Martiniani, S. P. Ong, I. Poltavsky, K. Schmidt, S. Takamoto,

A. P. Thompson, J. Westermayr, B. M. Wood, A practical guide to machine learning interatomic potentials – status and future, Current Opinion in Solid State and Materials Science 35 (2025) 101214. URL: http://dx.doi.org/10.1016/j.cossms.2025.101214. doi:10.1016/j.cossms.2025.101214.

[3] P. Pareti, G. Konstantinidis, T. J. Norman, M. Şensoy, SHACL constraints with inference rules, in: Lecture Notes in Computer Science, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2019, pp. 539–557.

[4] V. Venugopal, E. Olivetti, Matkg: An autonomously generated knowledge graph in material science, Scientific Data 11 (2024). URL: http://dx.doi.org/10.1038/s41597-024-03039-z. doi:10.1038/s41597-024-03039-z.

[5] Y. Zhang, F. Chen, Z. Liu, Y. Ju, D. Cui, J. Zhu, X. Jiang, X. Guo, J. He, L. Zhang, X. Zhang, Y. Su, A materials terminology knowledge graph automatically constructed from text corpus, Scientific Data 11 (2024). URL: http://dx.doi.org/10.1038/s41597-024-03448-0. doi:10.1038/s41597-024-03448-0.

[6] K. Ding, Z. Zhu, Y. Tang, K. Feng, X. Zhuang, H. Wang, Y. Yang, H. Du, Z. Ni, S. Wang, et al., Bridging data and discovery: a survey on knowledge graphs in ai for science, National Science Review 13 (2026) nwag140.

[7] X. Zhang, C. Zhao, X. Wang, A survey on knowledge representation in materials science and engineering: An ontological perspective, Computers in Industry 73 (2015) 8–22.

[8] E. Norouzi, J. Waitelonis, H. Sack, The landscape of ontologies in materials science and engineering: A survey and evaluation, arXiv preprint arXiv:2408.06034 (2024).

[9] V. Tshitoyan, J. Dagdelen, L. Weston, A. Dunn, Z. Rong, O. Kononova, K. A. Persson, G. Ceder, A. Jain, Unsupervised word embeddings capture latent knowledge from materials science literature, Nature 571 (2019) 95–98.

[10] L. Weston, V. Tshitoyan, J. Dagdelen, O. Kononova, A. Trewartha, K. A. Persson, G. Ceder, A. Jain, Named entity recognition and normalization applied to large-scale information extraction from the materials science literature, Journal of chemical information and modeling 59 (2019) 3692–3702.

[11] Y. Zhang, F. Chen, Z. Liu, Y. Ju, D. Cui, J. Zhu, X. Jiang, X. Guo, J. He, L. Zhang, et al., A materials terminology knowledge graph automatically constructed from text corpus, Scientific Data 11 (2024) 600.

[12] Y. Ye, J. Ren, S. Wang, Y. Wan, I. Razzak, B. Hoex, H. Wang, T. Xie, W. Zhang, Construction and application of materials knowledge graph in multidisciplinary materials science via large language model, Advances in Neural Information Processing Systems 37 (2024) 56878–56897.

[13] S. Li, S. Wei, C. Huang, Y. Zhang, G. Zhang, S. Sun, Extracting and reconstructing knowledge in materials science literature using large language models, Communications Materials (2026).

[14] G. C. Publio, D. Esteves, P. Panov, L. Soldatova, T. Soru, J. Vanschoren, H. Zafar, et al., Ml-schema: exposing the semantics of machine learning with schemas and ontologies, arXiv preprint arXiv:1807.05351 (2018).

[15] L. Kallab, E. Mansour, R. Chbeir, Sml: Semantic machine learning model ontology, in: International Conference on Web Information Systems Engineering, Springer, 2023, pp. 896–911.

[16] M. J. Luthfi, F. Darari, A. C. Ashardian, Sock: Shacl on completeness knowledge., in: WOP@ ISWC, 2022.