Scientific Knowledge Graph and Ontology Generation using Open Large Language Models

Alexandru Oarga[123] **Matthew Hart**[124]
Andres CM Bran[12] **Magdalena Lederbauer**[125]
Philippe Schwaller[12]
[1] Laboratory of Artificial Chemical Intelligence (LIAC), ISIC, EPFL
[2] National Centre of Competence in Research (NCCR) Catalysis, EPFL
[3] Department of Mathematics and Computer Science, University of Barcelona
[4] Department Applied Physical Sciences, UNC Chapel Hill
[5] Department of Chemistry and Applied Biosciences, ETH Zurich
philippe.schwaller@epfl.ch

Abstract

Knowledge Graphs (KGs) are powerful tools for structured information modeling, increasingly recognized for their potential to enhance the factuality and reasoning capabilities of Large Language Models (LLMs). However, in scientific domains, KG representation is often constrained by the absence of ontologies capable of modeling complex hierarchies and relationships inherent in the data. Moreover, the manual curation of KGs and ontologies from scientific literature remains a time-intensive task typically performed by domain experts. This work proposes a novel method leveraging LLMs for zero-shot, end-to-end ontology, and KG generation from scientific literature, implemented exclusively using open-source LLMs. We evaluate our approach by assessing its ability to reconstruct an existing KG and ontology of chemical elements and functional groups. Furthermore, we apply the method to the emerging field of Single Atom Catalysts (SACs), where information is scarce and unstructured. Our results demonstrate the effectiveness of our approach in automatically generating structured knowledge representations from complex scientific literature, in areas where manual curation is challenging or time-consuming. The generated ontologies and KGs provide a foundation for improved information retrieval and reasoning in specialized fields, opening new avenues for LLM-assisted scientific research and knowledge management. [1]

1 Introduction

Knowledge Graphs (KGs) are a powerful tool for representing structured information, enabling inference, retrieval, and analysis over large amounts of data. KGs model the real world in a graph representation, where nodes represent entities and edges represent relationships between them[13]. Recently, KGs have gained traction in the field of Natural Language Processing (NLP), demonstrating their potential to enhance the capabilities of Large Language Models (LLMs)[1,22,37]. It is well-established that LLMs are prone to 'hallucinations', where they generate incorrect or misleading information[51]. In this context, KGs have been shown to help mitigate this problem through Retrieval Augmented Generation (RAG), where graph structure is embedded as textual information as input to the LLM[17,27,42,50]. Moreover, KGs have also been shown to improve the reasoning capabilities of LLMs[32,49,52].

[1] Source code is available under MIT license at https://github.com/schwallergroup/ontorag/tree/main/src/OntoGen.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

Knowledge Graph Extraction (KGE) from text is usually performed through triplets extraction of the form (subject, predicate, object), where entities are extracted from the text using Named Entity Recognition (NER) and relationships between entities are inferred from the context[2]. Currently, LLMs are state-of-the-art in many NLP tasks, including NER[25,36], making them promising candidates for unsupervised KGE[20,35,46]. KGs based on triplets, however, often fail to model scientific fields such as bioinformatics or chemistry where more complex structures are needed[6,19,43]. Additionally, most KGE methods are based on the extraction of triplets from short sentences or paragraphs[5]. These methods are typically pre-trained on specific text corpora or domains or require labeled subsets of samples, which hampers their ability to generalize to different contexts[12,38]. All these factors limit the application of KGE to scientific literature, which remains considerably understudied[5]. In complex domains, the enrichment of KGs with ontologies can significantly improve information representation. Ontologies themselves are composed of classes, organized in a hierarchical structure denoted as taxonomy, containing relationships between classes. Each of the entities in the KGs can be considered as an instance of one or more classes in the ontology. Thus, ontologies provide a high-level schema for the knowledge present in the KG[13,14]. Ontologies however are commonly built and curated manually by domain experts, a process that is typically time-consuming and labor-intensive. Given the success of LLMs in many NLP tasks, some solutions have been proposed for ontology generation using LLMs. These methods, however, often have limitations: they may require pre-defined training sets for fine-tuning[8,34], cannot generate full ontologies[11], do not scale beyond smaller problems[3,21], or solely rely on the LLMs knowledge, making their application to scientific fields unpractical or unclear and hindering genralisation into other domains.[7,53].

There is a clear need in the field to generate hallucination-free knowledge extraction from scientific text in a way that is human-interpretable, consistent, and does not require efforts in fine-tuning language models on niche subjects. In this work, a method is proposed for end-to-end zero-shot ontology and KG generation from scientific literature using LLMs. To evaluate the performance, we measure the ability of the proposed method to reconstruct an existing KG and ontology of chemical elements. We then apply the method to the scientific domain of Single Atom Catalysis (SAC)[10,28,40,44], an emerging field in catalysis with no previously existing KG. In doing so, we create what is, to our knowledge, the first example of a domain-specific ontology for SAC. We demonstrate the value general-purpose LLMs add to the automation of the typically laborious process of ontology construction. The presented work therefore represents an advancement in the use of LLMs to not only perform domain-specific question answering but also extract domain-specific knowledge in a way that can be interpreted by experts. Finally, to demonstrate the utility of our work, we show that the combination of an ontology and a KG improves the quality of RAG-based solutions that rely only on KGs, such as GraphRAG[17].

2 Methods

2.1 Ontologies as a Form of Knowledge Representation

From a practical engineering perspective, ontologies are a specification of conceptualization using description logic. They model relationships between concepts and the words used to represent them. In their most basic form, an ontology can be thought of as a series of "isA" relationships formed in a hierarchical tree structure called a taxonomy. This taxonomy serves as the backbone of an ontology and can be engineered to include relationships, instances, properties, and axioms. For the purpose of this work, an ontology can be formally defined as follows:

Definition 2.1 *An* ontology *is a tuple* $\{\mathcal{C}, \mathcal{R}, \mathcal{I}, \mathcal{P}, \mathcal{A}\}$ *where:*

- $\mathcal{C}$ *is the set of all concepts present in the ontology such that* $\mathcal{C} = \{c_1, c_2, c_3, ..., c_n\}$
- $\mathcal{R}$ *is the set of all relationships present in the ontology such that* $R = \{(c_i, r_s, c_j) | c_i, c_j \in \mathcal{C} \ \& \ r_s \in \mathcal{R}_f\}$ *where* $\mathcal{R}_f$ *is the set of all possible relations*
- $\mathcal{I}$ *is the set of all instances of concepts present in the ontology such that* $\mathcal{I} = \{i_1, i_2, i_3, ..., i_n\}$ *and* $\forall(i)\exists(c)|i \in c$
- $\mathcal{P}$ *is the set of all possible properties in an ontology such that* $\mathcal{P} = \{p_1, p_2, p_3, ..., p_n\}$ *and* $p : \mathcal{I} \rightarrow \mathcal{V}$ *or* $p : \mathcal{C} \rightarrow \mathcal{V}$ *where* $\mathcal{V}$ *is the set of all possible values for a property*

- *and $\mathcal{A}$ is the set of axioms in the ontology.*

A key practical application of ontologies is the construction and integration of databases. We demonstrate the usefulness of our pipeline by using constructed ontologies to create graph databases, otherwise known as knowledge graphs. To relate knowledge graphs to ontologies, we utilize the following definition:

Definition 2.2 *A* knowledge graph *is set of tuples* $\{\mathcal{S}, \mathcal{P}, \mathcal{O}\}$ *where*

- *$\mathcal{S}$ is the set of subjects where $s \in \mathcal{S}$ is an instance of a concept from the ontology $=> \mathcal{S} \in \mathcal{I}$*
- *$\mathcal{P}$ is the set of predicates where $p \in \mathcal{P}$ is a relation from the ontology $=> p \in \mathcal{R}_f$*
- *$\mathcal{O}$ is the set of objects where $o \in \mathcal{O}$ is also an instance of a concept from the ontology such that $p : s \xrightarrow{p} o$*

From this definition, it follows that the ontology serves as the conceptual schema for the construction of a knowledge graph. The following methods detail our approach to leveraging zero-shot learning for creating an ontology of SAC and the subsequent construction of an ontology-informed knowledge graph.

Example 2.3 *Figure 1 shows an example ontology with four classes (white nodes) and a KG with three instances (shaded nodes). The ontology has three isA and is related to the KG through three isntanceOf relationships. Between the nodes of the KG, one relationship exists of the type 'supported on'. The term 'Thing' is always the root node of an ontology.*

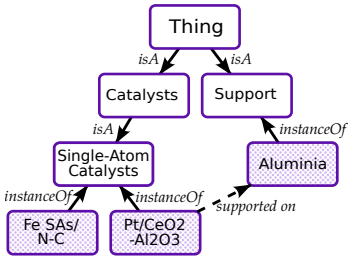

Figure 1: Illustrative example of an ontology and a knowledge graph in the domain of SAC.

2.2 In-Context Ontology Generation (ICOG)

Here, we introduce our method for In-Context Ontology Generation (ICOG) and KGE from scientific literature. The method is split into five separate steps: (1) the vocabulary that will be used to generate the ontology is extracted from the text, (2) the higher-level categories of the ontology are generated, (3) the taxonomy of the ontology is extracted, (4) the knowledge graph is instantiated from the extracted ontology, and (5) the relationships between the terms in the vocabulary are extracted.

Vocabulary Extraction The first step in our method is the extraction of all the key vocabulary terms from the text. The extracted terms serve as the base vocabulary that will compose the ontology. It follows that any terms that are not extracted in this step will be absent from the final ontology. Vocabulary extraction is itself split into three steps: (1) terms extraction, (2) acronyms extraction, and (3) lemmatization.

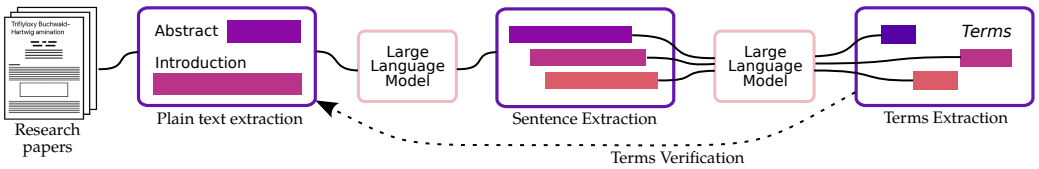

Figure 2: Illustration of the vocabulary extraction process.

Terms Extraction Traditional models trained for specific NER tasks often fail to generalize to domains not included in their training data. To address this limitation, we utilize open general-purpose LLMs for zero-shot vocabulary identification. The relevant text is first split into sentences to ensure individual words can be obtained in a fine-grained, detailed manner. This sentence splitting is performed using a pre-trained LLM that predicts sentence boundaries. Empirically, it was observed that splitting the text into smaller sentences allowed for a more fine-grained extraction and increased the number of terms extracted. For each of the extracted sentences, an LLM is then prompted to list

each of the terms present in the sentence. Finally, to avoid terms hallucinated by the LLM or subtle changes in the writing of the terms, a verification step is performed by ensuring that each of the listed terms is present in the original sentence.

Acronym Extraction Acronyms are commonly used in scientific literature to refer to long terms. To be able to associate each of the extracted terms of the previous step with their respective acronyms, we employ a separate procedure similar to terms extraction, but with the LLM prompted specifically to list all acronyms and their corresponding terms. As in the previous step, a verification procedure is used to ensure that the pairs of acronym terms listed are present in the original text.

Lemmatization To handle different writing forms of the same term (e.g. "Single-Atom Catalyst" and "Single Atom Catalysts"), we apply lemmatization to each extracted term after removing punctuation. This process reduces words to their root form, avoiding grammatical variations (e.g. "Single-Atom Catalysts" becomes "Single Atom Catalyst"). Terms that are mapped to the same root form after lemmatization are considered to refer to the same concept. This is particularly useful in the taxonomy generation process, where the same concept can appear in different forms across different papers. When such terms are identified, they are merged into a single term in the taxonomy.

Category extraction The first level of the taxonomic hierarchy, which is usually denoted as categories, consists of the main concepts that the ontology will cover. For instance, in the field of chemistry, the first level of the taxonomy might include categories such as elements, compounds, and reactions. However, these high-level categories cannot always be extracted from individual papers, as each scientific paper typically focuses on a particular area within a domain. Therefore, the highest level concepts must be extracted across several of the papers available to the system.

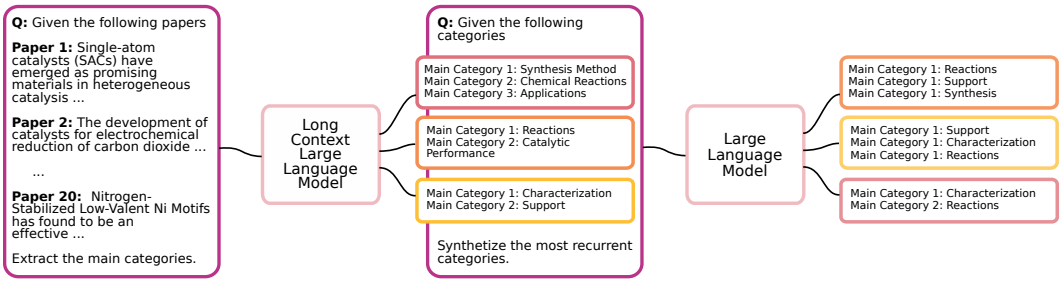

Figure 3: Illustration of the generation and refinement steps from the categories extraction process.

To accomplish this, we propose a two-step process:

1. *Generation:* Generating a list of categories from a set of papers is challenging for conventional LLMs due to their limited context window. To overcome this limitation, we employ long-context LLMs (LCLLMs)[9,24,29,39] for this task. First, a random sample of papers is presented to an LCLLM, which is then prompted to extract the main categories across the given papers. As LCLLMs are known to be sensitive to input order[30], we generate multiple answers, each with a random shuffle in the order of the papers.
2. *Refinement:* Using the answers generated in the previous step, we prompt an LLM to generate a curated list of categories using the most frequently occurring ones. Self-consistency is applied in this step to obtain categorical consistency. The definition of self-consistency is recalled in Appendix B.

It is important to note that the outcome of the previous step is a list of categories that will serve as the seed for the rest of the ontology. However, the correctness of the ontology depends on the downstream application that it is intended for. Therefore, some manual effort may be required at the end of this step to select the most relevant categories according to the specific goals of the user.

Taxonomy extraction The relationships in a taxonomy structure consist entirely of *isA* relationships. Once the vocabulary is extracted and the main categories are generated, the vocabulary needs to be organized into a hierarchical structure. Since the first level of the hierarchy has already been established, a top-down approach is the most logical method for generating the taxonomy. Although taxonomic information can be extracted from each paper separately, it must ultimately be consolidated

into a global unified taxonomy. Therefore, an incremental and sequential approach is proposed for constructing a global taxonomy.

Let $\mathcal{P} = \{P_1, P_2, ..., P_N\}$ be a corpus of N papers, $\mathcal{V} = \{V_1, V_2, ..., V_N\}$ a the set of terms extracted from the papers, where V_i is the vocabulary from paper P_i, and $\mathcal{T}^{(k)} = \{(s_1, t_1), (s_2, t_2), ..., (s_M, t_M)\}$ a taxonomy at iteration k, where each (s_i, t_i) with $s_i, t_i \in \mathcal{V}$ is a pair of terms that represents a *isA* relationship. The procedure to generate the full taxonomy is presented in Algorithm 1.

Algorithm 1 Iterative and Incremental Top-Down Taxonomy Generation

Input: Papers $\mathcal{P}$, Vocabulary $\mathcal{V}$, Initial Taxonomy $\mathcal{T}^{(0)}$
Output: Reconstructed Taxonomy after K iterations $\mathcal{T}^{(K)}$

for $k = 1, \ldots, K$ **do**
  $\mathcal{T}^{(k)} \leftarrow \mathcal{T}^{(k-1)}$
  for $P_i \in \mathcal{P}$ **do**
    $R_i \leftarrow$ query_relationships$(P_i, V_i, \mathcal{T}^{(k)})$
    for $(s, t) \in R_i$ **do**
      if is_valid$((s, t), \mathcal{T}^{(k)})$ **then**
        $\mathcal{T}^{(k)} \leftarrow \mathcal{T}^{(k)} \cup \{(s, t)\}$
return $\mathcal{T}^{(K)}$

In this algorithm, `query_relationships` extracts pairs (s, t) of *isA* relationships based on the information presented in paper P_i, where s is a term present in the taxonomy $\mathcal{T}^{(k)}$ and t is a term from the vocabulary V_i. The primary goal of `query_relationships` is to place each term into the existing taxonomy. Note that `query_relationships` may not necessarily return a single relationship for each term, but rather a subset of relationships. The `is_valid` function checks that no loop is created in the taxonomy if a given relationship is inserted. This procedure builds a taxonomy incrementally, requiring multiple iterations since a term may not be placed in the earlier taxonomies but may find its position in a later one.

In our implementation, `query_relationships` prompts an LLM with the content of the paper, the complete list of terms in the taxonomy, and the vocabulary terms to be queried. An example of such a prompt and its corresponding answer can be found in Appendix C. Given that the quality of the generated taxonomy heavily depends on the performance of this specific function, self-consistency is employed to mitigate the number of hallucinations generated, this is, for each query, many samples are generated and the majority voting of the answers is taken as final answer.

Knowledge Graph Instantiation Once the full taxonomy is constructed, the leaf nodes of the taxonomy are considered to be instances for the KG. We reason that since leaf nodes have no children, they should be treated as the lowest-level concepts in our knowledge representation schema, and thus can be assumed to be instances of the KG. Conversely, non-leaf nodes, having children related through *isA* relationships, are considered higher-level classes in the ontology rather than instances of the KG.

Relationship extraction Finally, we perform relationship extraction using a conventional triplet extraction approach. Specifically, we prompt an LLM to generate triplets from a given text, considering the extracted vocabulary. The resulting triplets are then filtered to retain only those involving terms from the vocabulary.

Table 1: Computational complexity of each step in ICOG and KGE in terms of LLM calls and prompt length.

Step	LLMs Calls	Prompt Length
Vocabulary Extraction	$O(N)$	$O(T)$
Categories Extraction	$O(S_G + S_R)$	$O(T \cdot N)$
Taxonomy Extraction	$O(K \cdot N)$	$O(T + \lvert\mathcal{T}\rvert)$
Relationships Extraction	$O(N)$	$O(T)$

The overall complexity of the method is reported in Table 1, where N is the number of papers, S_G and S_R are the number of samples in both steps of categories extraction, K is the number of iterations,

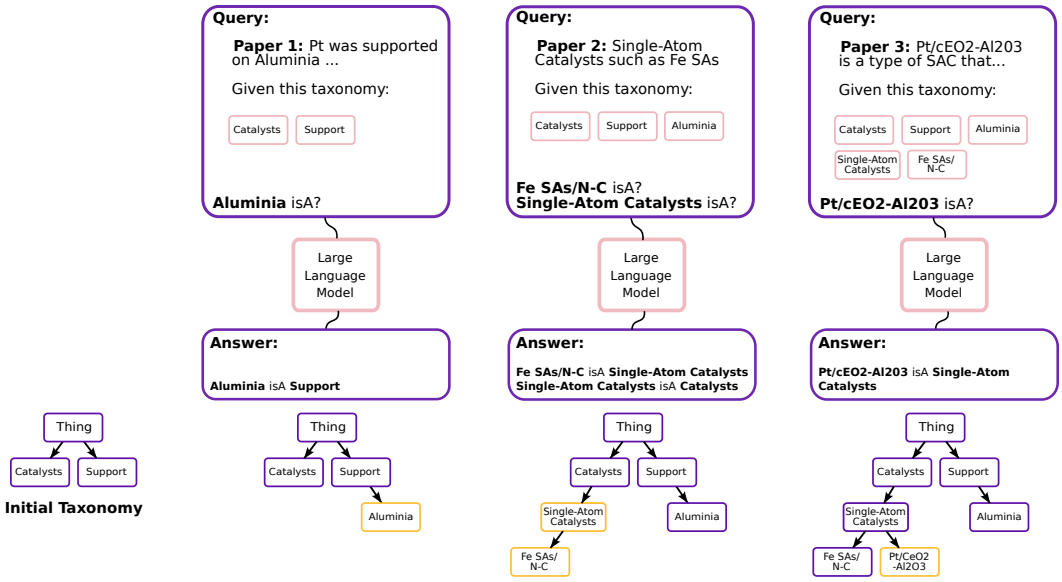

Figure 4: Illustration of the iterative and incremental taxonomy generation process.

T is the length of a paper and $\mathcal{T}$ is the size of a taxonomy. Notice that, except for the categories extraction step, only one paper at a time is processed in each LLM call.

Details on the packages and models used for the implementation of the pipeline are found in Appendix A

2.3 Ontology Reconstruction Evaluation

Evaluating a reconstructed ontology is not a trivial task, as the quality of the ontology is highly dependent on the downstream application for which it is intended. Additionally, many correct ontologies can be generated from the same text. To overcome the limitations of common metrics such as *term accuracy*, the percentage of the terms in the ground truth taxonomy that appear in the reconstructed taxonomy, and *knowledge graph accuracy*, the percentage of the instances in the ground truth KG that are also instances of the reconstructed KG, we propose the hierarchical accuracy as an evaluation metric for reconstructed ontologies.

Hierarchical Accuracy The main idea behind hierarchical accuracy is that if a node in the reconstructed taxonomy shares a common ancestor with a node in a ground truth taxonomy, then that node is considered to be correctly placed in the taxonomy. The hierarchical accuracy is then defined as the average number of nodes from the ground truth taxonomy that are correctly placed in the reconstructed taxonomy. This metric accounts for reconstructions that are more general than the ground truth, but penalizes reconstructions with missing nodes or misplaced nodes.

Generally, it is accepted that a term can appear more than once in a taxonomy. As a consequence, taxonomies are not modeled as trees but rather as Directed Acyclic Graphs (DAGs). Below, the hierarchical accuracy metric is formally defined.

Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be a DAG, where $\mathcal{N}$ is the set of nodes and $\mathcal{E}$ is the set of edges. The graph has a root node $r \in \mathcal{N}$, and every node $n_i \in \mathcal{N}$ is reachable from r. The set of all paths from r to any node n_i, is denoted $\mathcal{P}(r, n_i, \mathcal{G})$, and is defined as:

$$\mathcal{P}(r, n_i, \mathcal{G}) = \{(r, n_1, n_2, \ldots, n_i) \mid \mathcal{G} = (\mathcal{N}, \mathcal{E}), n_i \in \mathcal{N}, (n_j, n_{j+1}) \in \mathcal{E}\} \tag{1}$$

Let $\hat{\mathcal{G}} = (\hat{\mathcal{N}}, \hat{\mathcal{E}})$ be the reconstructed DAG from the text, which also contains the same root node $r \in \hat{\mathcal{N}}$. The *hierarchical accuracy* is then defined as:

$$\text{Hierarchical Accuracy} = \frac{1}{|\mathcal{N}|} \sum_{n_i \in \mathcal{N}} \text{CA}(n_i, \mathcal{G}, \hat{\mathcal{G}}) \tag{2}$$

where $\text{CA}(n_i, \mathcal{G}, \hat{\mathcal{G}})$ is a binary function defined as follows:

$$\text{CA}(n_i, \mathcal{G}, \hat{\mathcal{G}}) = \begin{cases} 1 & \text{if } n_i \in \hat{\mathcal{N}} \text{ and } \left(\text{NP}(r, n_i, \mathcal{G}) \cap \text{NP}(r, n_i, \hat{\mathcal{G}})\right) \setminus \{r\} \neq \varnothing \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Here, $\text{NP}(r, n_i, \mathcal{G})$ represents the set of all nodes appearing on any path from r to n_i in $\mathcal{G}$:

$$\text{NP}(r, n_i, \mathcal{G}) = \bigcup_{p \in \mathcal{P}(r, n_i, \mathcal{G})} \{n | n \in p\} \tag{4}$$

In other words, $\text{CA}(n_i, \mathcal{G}, \hat{\mathcal{G}})$ is equal to 1 if node n_i has at least one common ancestor (other than the root node r) in both the ground truth taxonomy $\mathcal{G}$ and the reconstructed taxonomy $\hat{\mathcal{G}}$. CA does not count the root node r as a common ancestor as this term is common to all terms in both DAGs. The hierarchical accuracy is then the average over all nodes in the ground truth taxonomy.

3 Results & Discussion

Here we present the results obtained with the proposed method for ontology and knowledge graph generation from scientific literature. The performance of the method is first evaluated by measuring the ability to reconstruct an existing KG and ontology of chemical elements and functional groups from a corpus of text. The method is then applied to construct the first KG and ontology of the scientific field of SAC. The effectiveness of the ontology is then evaluated by applying it in a RAG-based solution for SACs domain.

3.1 Chemical Elements and Functional Groups Reconstruction

ElementKG[18] is an ontology and KG of chemical elements and functional groups with 272 nodes, of which 188 are instances. This ontology and KG were curated manually by domain experts from information primarily sourced from Wikipedia. To reconstruct ElementKG, 76 Wikipedia pages corresponding to functional groups (e.g. Alkyne, Alkane, etc.) and groups of elements (e.g. Noble gases, Metalloids, etc.) were collected for our work. Except for the functional groups page, only the introduction section of the pages was used for the reconstruction, as it was observed to contain most of the relevant taxonomical information. Additionally, four plain texts were manually curated for cases where no Wikipedia page exists (e.g. ferrous elements) or the information available is not enough to reconstruct ElementKG (e.g. rare metals).

Initially, the extracted vocabulary contained 1735 terms, with an accuracy of 94.30% compared to ElementKG vocabulary. A total of 555 relationships between terms were extracted from the corpus. The evaluation metrics (term accuracy, hierarchical accuracy, and KG accuracy) for the reconstruction of ElementKG are presented in Table 2. The table also reports the metrics for each of the two main categories in ElementKG, elements and functional groups. To account for terms that might be repeated in a taxonomy, repeated terms in the ground truth taxonomy are treated as separate terms.

Table 2: Evaluation metrics for the reconstruction of ElementKG averaged over 2 runs. Overall accuracy (all) and the more finegrained analysis on the element (E) and functional groups (FG) categories.

Evaluation	Iteration 1	Iteration 2	Iteration 3	Iteration 4
Term acc. (all)	0.835 ± 0.108	0.889 ± 0.073	0.898 ± 0.060	0.907 ± 0.052
Term acc. (E)	0.934 ± 0.030	0.952 ± 0.026	0.952 ± 0.026	0.955 ± 0.032
Term acc. (FG)	0.626 ± 0.320	0.659 ± 0.325	0.681 ± 0.293	0.693 ± 0.278
Hierarchical acc. (all)	0.842 ± 0.110	0.868 ± 0.102	0.876 ± 0.089	0.889 ± 0.075
Hierarchical acc. (E)	0.966 ± 0.007	0.965 ± 0.017	0.965 ± 0.017	0.976 ± 0.007
Hierarchical acc. (FG)	0.647 ± 0.291	0.724 ± 0.234	0.746 ± 0.203	0.761 ± 0.182
KG acc. (all)	0.749 ± 0.068	0.727 ± 0.000	0.709 ± 0.011	0.706 ± 0.015
KG acc. (E)	0.900 ± 0.053	0.902 ± 0.020	0.902 ± 0.020	0.911 ± 0.007
KG acc. (FG)	0.450 ± 0.301	0.394 ± 0.151	0.363 ± 0.159	0.344 ± 0.133

As shown in Table 2, our method demonstrates high accuracy in reconstructing the Element taxonomy. The three metrics considered consistently exceed 90% across all iterations, indicating that the initial iteration provides a robust foundation for the reconstruction process. However, the accuracy for

functional groups is comparatively lower. Notably, both term accuracy and hierarchical accuracy for functional groups show improvement over successive iterations, suggesting that new terms are being correctly integrated into the taxonomy structure. Conversely, the KG accuracy tends to decrease after the second iteration, implying that some terms are incorrectly positioned beneath nodes that should be leaf instances in the KG—i.e., terms that should not have any child nodes. We observed considerable variability in the functional groups' performance across different runs, attributable to the inherent stochasticity of LLMs. This variability in the initial iteration significantly influenced the performance of subsequent iterations, highlighting the sensitivity of the process to initial conditions.

Figure 5 shows general descriptive statistics of the reconstructed taxonomy. The figures illustrate how the number of *isA* relationships increases constantly over iterations. We observed that the root cause of this is general terms (e.g. Materials) that are placed in the taxonomy, and that, given their general nature, cause a large number of terms to be placed below them. Further statistics related to the reconstruction can be found in Appendix D.

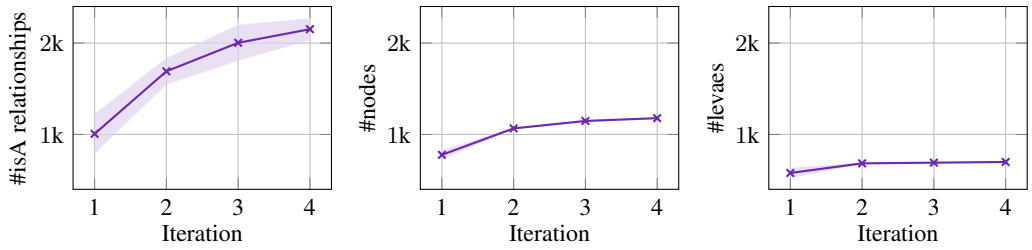

Figure 5: Mean number of *isA* relationships, nodes and leaves (blue lines) and standard deviation (shaded areas) averaged over two runs.

3.2 Ontology and KG generation for Single Atom Catalysis

SAC, a recent scientific domain studying the catalytic properties of isolated single metal atoms[28,40], presents an ideal case study for our method due to its mostly unstructured literature. We applied our procedure to generate the first ontology and KG of this field, using a corpus of 20 papers. The papers were carefully selected to cover different synthesis and characterization methods of SACs. Focusing on the abstract and introduction sections, where concepts are typically introduced and explained, we extracted 1944 terms with 296 relationships. The category identification process initially yielded 331 categories, refined to 131, and manually curated to 12 key categories. Figure 6 shows the evolution of the number of nodes, *isA* relationships and leaves over iterations. Further details are provided in Appendix D.

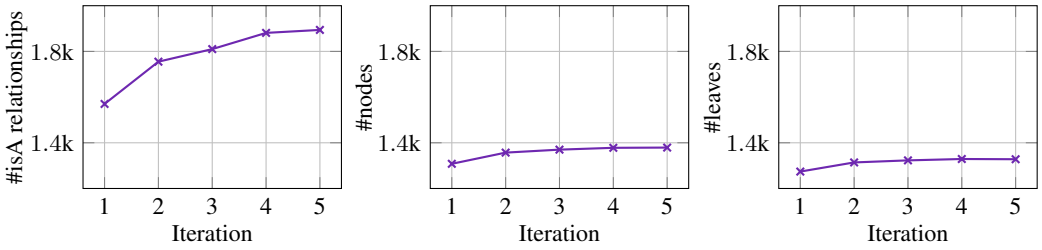

Figure 6: Number of nodes, *isA* realtionships and leaves of the generated SACs taxonomy over iterations.

After five iterations, the growth in nodes, *isA* relationships, and leaves slowed significantly, indicating convergence in taxonomy generation, despite not exhausting the entire vocabulary. This suggests the method's effectiveness in structuring knowledge from an emerging scientific field.

GraphRAG-based downstream evaluation To demonstrate the practical value of our generated ontology, we conducted a comparative study in the SAC domain. We compared two approaches: a

standard KG approach and an enhanced KG+Ontology approach, both integrated into a Retrieval-Augmented Generation (RAG) solution based on the GraphRAG procedure [17].

The GraphRAG pipeline is composed of the following steps: (1) the generation of the KG from plain text, (2) description generation for each term and relationship, (3) clustering for community generation, (4) community description generation, and (5) community retrieval for answer generation. In our enhanced approach, we extended the KG in step (1) with our generated ontology (KG+ontology).

To evaluate performance, we used the same four metrics as the original GraphRAG study [17]: comprehensiveness, diversity, empowerment, and directness. We created a dataset of 50 questions using long-context LLMs, covering various SAC topics such as synthesis, characterization methods, and applications, which were then validated by domain experts.

Following the GraphRAG methodology [17], we used an LLM, in our work Anthropic's Claude 3.5 Sonnet, to evaluate the quality of generated answers for both approaches. Table 3 presents the results, clearly showing that our KG+Ontology approach significantly outperforms the standard KG-only method across all metrics. This improvement demonstrates the ontology's ability to enhance term relationships, enabling better clustering and retrieval, ultimately resulting in more informative and higher-quality answers.

Table 3: Win-rate according to different criteria in the GraphRAG setting with a dataset of 50 questions.

Model	**Comprehensiveness**	**Diversity**	**Empowerment**	**Directness**
GraphRAG (KG)	1/50	4/50	1/50	16/50
GraphRAG (KG + Ontology)	49/50	46/50	49/50	34/50

Expert Evaluation To assess the quality of the generated ontology, a panel of experts conducted an evaluation of the taxonomical relationships. The experts randomly sampled relationships from various iterations of the ontology. Their task was to determine whether each sampled relationship was correct within the context of the domain. The results of this evaluation showed that, on average, 64.5% of the examined relationships were deemed correct by the experts. While this indicates a majority of accurate relationships, it also suggests room for improvement in the ontology generation process. Upon analysis of the incorrect relationships, the experts did not identify any clear patterns of errors. However, they provided three key observations for potential enhancement:

- *Specificity:* The quality of relationships could be improved by making them more specific, thereby reducing ambiguity and increasing precision.
- *Context Dependency:* Some extracted vocabulary terms are heavily context-dependent and may lose their intended meaning when placed in a more general context. This suggests a need for better context preservation or clearer domain boundaries.
- *Semantic Redundancy:* The taxonomy contains instances of semantically similar concepts repeated in different parts of the structure. This redundancy could be addressed to streamline the ontology.

These expert insights provide valuable direction for refining the ontology generation process and improving the overall quality of the taxonomical relationships.

4 Conclusion

Extraction of structured knowledge from scientific literature is a challenging task that is usually performed manually by domain experts. Given the success of LLMs in NLP tasks, this work proposes the use of LLMs as an information extraction engine from the scientific literature. A five-step pipeline is proposed that is able to generate ontologies and KGs with an in-context nature and without the need for any training data. The pipeline is then implemented entirely using openly available LLMs and long-context LLMs. The evaluation of the approach is done by reconstructing an existing KG and ontology of chemical elements and functional groups from the literature. The results show good accuracy in reconstructing the taxonomies and KGs with a term accuracy and hierarchical accuracy of over 80% in all iterations, and knowledge graph accuracy of over 70% in all iterations. It was observed, however, that the quality of the reconstruction is dependent on the first iteration generation, and that given the stochastic nature of the LLMs, there is variability in the results. The method was

then applied to the emerging domain of SACs. Given the recent development of the field, the literature lacks structured information, which makes the generation of an ontology and KG a challenging task. It was shown that the in-context generated ontology of SACs was able to improve the performance of the GraphRAG pipeline in question-answering tasks related to SACs. Finally, the quality was evaluated by a group of experts, who evaluated that 64.5% of the relationships were correct. We can thus conclude that the proposed approach is able to extract meaningful information from scientific literature.

References

[1] Garima Agrawal, Tharindu Kumarage, Zeyad Alghami, and Huan Liu. Can knowledge graphs reduce hallucinations in llms?: A survey. *arXiv preprint arXiv:2311.07914*, 2023.

[2] Tareq Al-Moslmi, Marc Gallofré Ocaña, Andreas L Opdahl, and Csaba Veres. Named entity extraction for knowledge graphs: A literature overview. *IEEE Access*, 8:32862–32881, 2020.

[3] Hamed Babaei Giglou, Jennifer D'Souza, and Sören Auer. Llms4ol: Large language models for ontology learning. In *International Semantic Web Conference*, pages 408–427. Springer, 2023.

[4] Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. Nougat: Neural optical understanding for academic documents. *arXiv preprint arXiv:2308.13418*, 2023.

[5] Andres M Bran, Zlatko Jončev, and Philippe Schwaller. Knowledge graph extraction from total synthesis documents. In *Proceedings of the 1st Workshop on Language+ Molecules*, pages 74–84, 2024.

[6] Emanuele Cavalleri, Alberto Cabri, Mauricio Soto-Gomez, Sara Bonfitto, Paolo Perlasca, Jessica Gliozzo, Tiffany J Callahan, Justin Reese, Peter N Robinson, Elena Casiraghi, et al. An ontology-based knowledge graph for representing interactions involving rna molecules. *Scientific Data*, 11(1):906, 2024.

[7] Boqi Chen, Fandi Yi, and Dániel Varró. Prompting or fine-tuning? a comparative study of large language models for taxonomy construction. In *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion*, pages 588–596. IEEE, 2023.

[8] Jiaoyan Chen, Yuan He, Yuxia Geng, Ernesto Jiménez-Ruiz, Hang Dong, and Ian Horrocks. Contextual semantic embeddings for ontology subsumption prediction. *World Wide Web*, 26(5):2569–2591, 2023.

[9] Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023.

[10] Zupeng Chen, Evgeniya Vorobyeva, Sharon Mitchell, Edvin Fako, Manuel A Ortuño, Núria López, Sean M Collins, Paul A Midgley, Sylvia Richard, Gianvito Vilé, et al. A heterogeneous single-atom palladium catalyst surpassing homogeneous systems for suzuki coupling. *Nature nanotechnology*, 13(8):702–707, 2018.

[11] Giovanni Ciatto, Andrea Agiollo, Matteo Magnini, and Andrea Omicini. Large language models as oracles for instantiating ontologies with domain-specific knowledge. *arXiv preprint arXiv:2404.04108*, 2024.

[12] John Dagdelen, Alexander Dunn, Sanghoon Lee, Nicholas Walker, Andrew S Rosen, Gerbrand Ceder, Kristin A Persson, and Anubhav Jain. Structured information extraction from scientific text with large language models. *Nature Communications*, 15(1):1418, 2024.

[13] Xin Luna Dong. Generations of knowledge graphs: The crazy ideas and the business impact. *arXiv preprint arXiv:2308.14217*, 2023.

[14] Xin Luna Dong, Xiang He, Andrey Kan, Xian Li, Yan Liang, Jun Ma, Yifan Ethan Xu, Chenwei Zhang, Tong Zhao, Gabriel Blanco Saldana, et al. Autoknow: Self-driving knowledge collection for products of thousands of types. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2724–2734, 2020.

[15] Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023.

[16] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[17] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

[18] Yin Fang, Qiang Zhang, Ningyu Zhang, Zhuo Chen, Xiang Zhuang, Xin Shao, Xiaohui Fan, and Huajun Chen. Knowledge graph-enhanced molecular contrastive learning with functional prompt. *Nature Machine Intelligence*, 5(5):542–553, 2023.

[19] Feroz Farazi, Jethro Akroyd, Sebastian Mosbach, Philipp Buerger, Daniel Nurkowski, Maurin Salamanca, and Markus Kraft. Ontokin: An ontology for chemical kinetic reaction mechanisms. *Journal of Chemical Information and Modeling*, 60(1):108–120, 2019.

[20] Johannes Frey, Lars-Peter Meyer, Natanael Arndt, Felix Brei, and Kirill Bulert. Benchmarking the abilities of large language models for rdf knowledge graph creation and comprehension: How well do llms speak turtle? *arXiv preprint arXiv:2309.17122*, 2023.

[21] Maurice Funk, Simon Hosemann, Jean Christoph Jung, and Carsten Lutz. Towards ontology construction with language models. *arXiv preprint arXiv:2309.09898*, 2023.

[22] Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. Towards foundation models for knowledge graph reasoning. *arXiv preprint arXiv:2310.04562*, 2023.

[23] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[24] Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. Lm-infinite: Zero-shot extreme length generalization for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3991–4008, 2024.

[25] Yan Hu, Qingyu Chen, Jingcheng Du, Xueqing Peng, Vipina Kuttichi Keloth, Xu Zuo, Yujia Zhou, Zehan Li, Xiaoqian Jiang, Zhiyong Lu, et al. Improving large language models for clinical named entity recognition via prompt engineering. *Journal of the American Medical Informatics Association*, page ocad259, 2024.

[26] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*, 2023.

[27] Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*, 2023.

[28] Selina K Kaiser, Zupeng Chen, Dario Faust Akl, Sharon Mitchell, and Javier Pérez-Ramírez. Single-atom catalysts across the periodic table. *Chemical reviews*, 120(21):11703–11809, 2020.

[29] Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context. *arXiv preprint arXiv:2310.01889*, 2023.

[30] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

[31] Yinhan Liu. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[32] Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. *arXiv preprint arXiv:2310.01061*, 2023.

[33] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2024.

[34] Patricia Mateiu and Adrian Groza. Ontology engineering with large language models. In *2023 25th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pages 226–229. IEEE, 2023.

[35] Igor Melnyk, Pierre Dognin, and Payel Das. Knowledge graph generation from text. *arXiv preprint arXiv:2211.10511*, 2022.

[36] Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56(2):1–40, 2023.

[37] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[38] Andrea Papaluca, Daniel Krefl, Sergio Mendez Rodriguez, Artem Lensky, and Hanna Suominen. Zero-and few-shots knowledge graph triplet extraction with large language models. *arXiv preprint arXiv:2312.01954*, 2023.

[39] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*, 2023.

[40] Botao Qiao, Aiqin Wang, Xiaofeng Yang, Lawrence F Allard, Zheng Jiang, Yitao Cui, Jingyue Liu, Jun Li, and Tao Zhang. Single-atom catalysis of co oxidation using pt1/feo x. *Nature chemistry*, 3(8):634–641, 2011.

[41] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[42] Priyanka Sen, Sandeep Mavadia, and Amir Saffari. Knowledge graph-augmented language models for complex question answering. In *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations*, pages 1–8, 2023.

[43] Kent A Shefchek, Nomi L Harris, Michael Gargano, Nicolas Matentzoglu, Deepak Unni, Matthew Brush, Daniel Keith, Tom Conlin, Nicole Vasilevsky, Xingmin Aaron Zhang, et al. The monarch initiative in 2019: an integrative data and analytic platform connecting phenotypes to genotypes across species. *Nucleic acids research*, 48(D1):D704–D715, 2020.

[44] Manu Suvarna, Alain Claude Vaucher, Sharon Mitchell, Teodoro Laino, and Javier Pérez-Ramírez. Language models and protocol standardization guidelines for accelerating synthesis planning in heterogeneous catalysis. *Nature Communications*, 14(1):7964, 2023.

[45] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1):1–12, 2019.

[46] Milena Trajanoska, Riste Stojanov, and Dimitar Trajanov. Enhancing knowledge graph construction using large language models. *arXiv preprint arXiv:2305.04676*, 2023.

[47] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[48] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[49] Yilin Wen, Zifeng Wang, and Jimeng Sun. Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models. *arXiv preprint arXiv:2308.09729*, 2023.

[50] Yike Wu, Nan Hu, Guilin Qi, Sheng Bi, Jie Ren, Anhuan Xie, and Wei Song. Retrieve-rewrite-answer: A kg-to-text enhanced llms framework for knowledge graph question answering. *arXiv preprint arXiv:2309.11206*, 2023.

[51] Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. Hallucination is inevitable: An innate limitation of large language models. *arXiv preprint arXiv:2401.11817*, 2024.

[52] Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. *arXiv preprint arXiv:2104.06378*, 2021.

[53] Qingkai Zeng, Yuyang Bai, Zhaoxuan Tan, Shangbin Feng, Zhenwen Liang, Zhihan Zhang, and Meng Jiang. Chain-of-layer: Iteratively prompting large language models for taxonomy induction from limited examples. *arXiv preprint arXiv:2402.07386*, 2024.

A Technical Stack and Experimental Setting

The following technical stack was used for the pipeline implementation:

- *Plain text extraction*: Plain text was extracted from the paper's PDF files using Nougat[4] with model `0.1.0-base`. When the extraction with Nougat failed, extraction resorted to PyMuPDF [2].
- *Terms Extraction*: The split of the initial text into sentences is done with a RoBeRTa base model[31] from the Spacy suite [3]. Multiple sentences are then concatenated as long as they don't exceed a maximum length of 2048 characters. Vocabulary extraction was performed with a Llama 3.1 70B model[16] quantized to 4 bits, with a maximum context length of 1024 tokens. It was observed that a greedy generation (temperature=0) provided the best results. For the other parameters, we set $k = 5$.
- *Category generation*: The *generation* step is performed with Llama-3 8B Gradient Instruct 1048k quantized to 8 bits, context length limit of 128K tokens and temperature=0.7. The *refinement* step was done with Llama-3.1 70B quantized to 4 bits, context limit of 32K tokens and temperature=0.5. After each of the two steps, an extra prompt is performed with the former model to format the output of the LLMs. A total of 80 different samples were generated both in the generation and refinement steps.
- *Taxonomy generation and relationships extraction*: Both taxonomy generation and relationships extraction are done with a Llama 3.1 70B model[16] quantized to 4 bits, with a maximum context length of 32K tokens and temperature=0.1. For each LLM query, self-consistency was applied by taking the majority voting of 3 sampled answers.

For the GraphRAG setting, the following hyperparameters were used:

- *Text split and vectorization:* Chunk length of 1200 characters with a 100 characters overlap. Text embeddings were computed using the `all-MiniLM-L6-v2` pretrained model from Sentence Trasnformers framework[41].
- *Node2vec[23]:* Dimensionality of 1536, with 10 walks of length 40 and window size 2.
- *Clustering:* Clustering was performed with the Leiden algorithm[45] as in the original GraphRAG work. Maximum graph cluster size of 10 nodes.
- *LLM evaluation:* The LLM evaluation of the answers was done using `claude-3-5-sonnet-20240620`.

LLM inference was done using the Ollama suite [4]. All the technical decisions made were adjusted to a budget of a single NVIDIA RTX A6000 graphics unit with 48GB. The source code is publicly available under MIT Licence.

[2] `github.com/pymupdf/PyMuPDF`

[3] `github.com/explosion/spaCy`

[4] `github.com/ollama/ollama`

B Self consistency

The improvement of LLMs' capabilities to generate high-quality, hallucination-free answers is currently a highly active area of research. Many generic methods have been proposed that improve LLMs outputs without training data, fine-tuning or reinforcement learning, which includes, among others, self-consistency[47], debating LLMs[15], and self-refinement[33]. Research by Huang et al.[26] demonstrates that self-consistency offers competitive results while being more computationally efficient compared to other methods. Therefore, in this work, self-consistency is used to improve the quality of answers from a LLM. As utilized in our approach, self-consistency can be defined as:

Definition B.1 *Let $a_1, a_2, ..., a_n \in \mathbb{A}$ be the answers to a given prompt p generated by a LLM, and r_i the set of tokens generated before the answer a_i.*

Self-Consistency *(SC) applies a marginalization over r_i by taking the majority vote of the answers a_i, i.e. $a = \arg\max_{a_i} \sum_{j=1}^{n} \mathbb{1}(a_i = a_j)$, thus giving as a final answer the most "consistent" answer generated by the LLM.*

It is important to note that self-consistency was initially proposed to enhance Chain of Thought (CoT) reasoning[48] in LLMs[47], to improve performance on generalized problem-solving tasks. In our work, we leverage the generalizability of self-consistency to improve the quality of our knowledge schemas reconstruction.

C Prompt sample

This Section includes a sample question and answer obtained from the taxonomy generation phase. Note that in the question below, given the restriction in the context length, the whole taxonomy cannot be included in the prompt. As a consequence, only the term and the main category it belongs (in parenthesis) are included. Notice also that, in the answer provided, not all the terms as classified, as some of them are classified as 'None'.

Sample Question

Given this context:
===
The actinide () or actinoid () series encompasses at least the 14 metallic chemical elements in the 5f series, with atomic numbers from 89 to 102, actinium through nobelium. (Number 103, lawrencium, is sometimes also included despite being part of the 6d transition series.) The actinide series derives its name from the first element in the series, actinium. The informal chemical symbol An is used in general discussions of actinide chemistry to refer to any actinide.
All the actinides are f-block elements. Lawrencium is sometimes considered one as well, despite being a d-block element and a transition metal. The series mostly corresponds to the filling of the 5f electron shell, although as isolated atoms in the ground state many have anomalous configurations involving the filling of the 6d shell due to interelectronic repulsion. In comparison with the lanthanides, also mostly f-block elements, the actinides show much more variable valence. They all have very large atomic and ionic radii and exhibit an unusually large range of physical properties. While actinium and the late actinides (from curium onwards) behave similarly to the lanthanides, the elements thorium, protactinium, and uranium are much more similar to transition metals in their chemistry, with neptunium, plutonium, and americium occupying an intermediate position.
All actinides are radioactive and release energy upon radioactive decay; naturally occurring uranium and thorium, and synthetically produced plutonium are the most abundant actinides on Earth. These have been used in nuclear reactors, and uranium and plutonium are critical elements of nuclear weapons. Uranium and thorium also have diverse current or historical uses, and americium is used in the ionization chambers of most modern smoke detectors. Of the actinides, primordial thorium and uranium occur naturally in substantial quantities.
===
,and given the following taxonomy:
===
Carbon (Element of the periodic table)
Element of the periodic table
Hemiacetals (Functional group)
R2C=O (Functional group)
Aldehyde hydrate (Functional group)
Carbonyl (Functional group)
Aldehyde (Functional group)
Mixed acetal (Functional group)
Ketone hydrate (Functional group)
Carbonyl compounds (Functional group)
Acetal (Functional group)
Esters (Functional group)
Ketone-derived structures (Functional group)
Symmetric acetal (Functional group)

Orthoester (Functional group)
Hydroxyl group (Functional group)
Ketal (Functional group)
Functional group
===

Complete the following list to classify the terms into the taxonomy according to the context.
If a term does not fit in any of the categories, say "None".
If the text is not clear enough to classify a term, say "None".
If the text does not explicitly mention that a term is a type of another, say "None".
If multiple categories apply, choose the most specific one.
In the ouput include only the classification. Do not include any explanation or additional information.
Do not classify a term to be its own parent, this is, do not output answers such as "A isA A".
The answer should include 'isA'.

Nuclear weapons isA
Lawrencium isA
F-block elements isA
Thorium isA
Radioactive decay isA
Actinides isA
Actinoid isA
Neptunium isA
Actinium isA
Actinide isA
Curium isA
Ground state isA
Primordial thorium isA
Americium isA
6d transition series isA
Plutonium isA
Protactinium isA
Lanthanides isA
Atomic numbers isA
Smoke detectors isA
Nuclear reactors isA
Ionic radii isA
Uranium isA
Electron shell isA
Transition metals isA
D-block element isA
Ionization chambers isA
5f series isA
Valence isA
Interelectronic repulsion isA
Nobelium isA
Metallic chemical elements isA

Sample Answer

Here is the list of classifications:

- Nuclear weapons isA None
- Lawrencium isA Actinide
- F-block elements isA None
- Thorium isA Actinide
- Radioactive decay isA None
- Actinides isA Metallic chemical elements
- Actinoid isA Actinide
- Neptunium isA Actinide
- Actinium isA Actinide
- Actinide isA F-block elements
- Curium isA Actinide
- Ground state isA None
- Primordial thorium isA Thorium
- Americium isA Actinide
- 6d transition series isA Transition metal
- Plutonium isA Actinide
- Protactinium isA Actinide
- Lanthanides isA F-block elements
- Atomic numbers isA None
- Smoke detectors isA None
- Nuclear reactors isA None
- Ionic radii isA None
- Uranium isA Actinide
- Electron shell isA None

- Transition metals isA Metallic chemical elements
- D-block element isA Transition metal
- Ionization chambers isA None
- 5f series isA F-block elements
- Valence isA None
- Interelectronic repulsion isA None
- Nobelium isA Actinide
- Metallic chemical elements isA Element of the periodic table

D Taxonomical properties

This Sections includes details of the taxonomical properties obtained with the two taxonomies reconstructed in this work. Table 4 includes the resulting taxonomical properties obtained for ElementKG reconstruction. Note that the original ElementKG is considerably smaller in comparison, as this taxonomy has been narrowed down to only the required terms, while this work reconstructs a general-purpose taxonomy. Table 5 presents the main taxonomical properties of the SACs taxonomy.

Table 4: Taxonomical properties of the reconstructed ElementKG across iterations averaged over 2 runs.

Property	**ElementKG**	**Iteration 1**	**Iteration 2**	**Iteration 3**	**Iteration 4**
#nodes	272	777.0 ± 53.7	1066.5 ± 3.5	1147.5 ± 10.6	1178.5 ± 0.7
#isA	347	1007.0 ± 217.8	1691.0 ± 144.2	2003.0 ± 196.6	2151.5 ± 119.5
#leaves	188	578.5 ± 54.4	684.0 ± 11.3	691.5 ± 2.1	699.0 ± 1.4
Depth	8	17.0 ± 2.8	27.5 ± 9.2	31.5 ± 10.6	37.0 ± 9.8
Max #isA per node	32	134.0 ± 97.6	139.5 ± 98.3	145.0 ± 93.3	148.0 ± 89.0
Mean #isA per node	1.28	1.28 ± 0.19	1.58 ± 0.14	1.74 ± 0.15	1.83 ± 0.10
Mean steps to leaf	3.63	3.69 ± 0.43	4.13 ± 0.57	4.19 ± 0.50	4.20 ± 0.47

Table 5: Taxonomical properties of the generated SACs taxonomy.

Property	**Iteration 1**	**Iteration 2**	**Iteration 3**	**Iteration 4**	**Iteration 5**
#nodes	1308	1357	1370	1378	1379
#isA	1574	1755	1810	1881	1894
#leaves	1274	1314	1323	1329	1328
Mean #isA per node	1.1888	1.2711	1.285	1.3091	1.3161
Max #isA per node	349	367	378	385	386
Depth	5	6	6	6	6
Mean steps to leaf	2.0259	1.9955	1.9956	1.9956	1.9956