Verifiable Knowledge Graph Construction for Materials Synthesis: A Hybrid Domain Model–LLM–Expert Information Extraction Framework

Anonymous ACL submission

Abstract

Materials synthesis procedures contain high-value scientific knowledge, but are largely trapped in the unstructured literature. Extracting such knowledge is challenging because the synthesis text is low-resource and schema-sensitive, making existing LLM-only or generic information extraction approaches unreliable. We present VKGC, an incremental three-stage framework for verifiable materials synthesis information extraction that combines a domain-tuned generative model, LLM-based correction, and expert validation. Using this framework, we construct an expert-level materials synthesis knowledge graph and a benchmark for synthesis-route planning. Experiments show that the domain-tuned model achieves high precision even with limited annotations, while LLM-based correction substantially improves recall, especially for relations and event arguments. We further show that frontier LLMs perform poorly on the synthesis planning benchmark despite high self-confidence, highlighting the gap between general language competence and domain-grounded procedural reasoning. Our results suggest that in low-resource scientific domains, reliable knowledge graph construction requires hybrid human–AI pipelines rather than LLM-only extraction, and we release a benchmark to support future work on scientific information extraction and materials synthesis planning.

1 Introduction

1.1 Background

Knowledge graphs have emerged as a powerful framework for modeling complex data. By representing information as a directed graph, where nodes denote entities and edges represent the semantic relationships between them, knowledge graphs provide a structured and machine-readable form. Over the past decade, knowledge graphs have become ubiquitous in academia, driving significant advancements in downstream applications such as question answering, recommendation systems, semantic search, and natural language processing. Constructing knowledge graphs requires extracting structured facts from unstructured text via Named Entity Recognition (NER), Relation Extraction (RE), and Event Extraction (EE). Because event extraction identifies trigger words and their arguments, it captures highly structured relational information essential for scientific domains. The coverage and precision of the underlying NER system determine the quality of the constructed knowledge graph. Errors or omissions during the entity recognition phase propagate through the pipeline, leading to incomplete or erroneous graphs.

Despite widespread interest and proven efficacy, the construction and maintenance of knowledge graphs are severely limited by their cost. The construction of knowledge graphs in real-world applications is notoriously expensive. The current state-of-the-art NER model is not yet capable of constructing knowledge graphs without additional help, as the extracted entities, relations, and events are incomplete and noisy. Besides, these NER models often require additional training when transferring knowledge to specific domains. To construct reliable knowledge graphs, manual curation by domain experts is required. Manual curation guarantees high reliability, but it also means high costs in terms of time, labor, and financial resources, making it hardly scalable in the construction of knowledge graphs.

1.2 Motivations

The traditional approach to materials synthesis relies heavily on trial and error. Decades of research have created knowledge trapped in the unstructured text of millions of scientific publications. Synthesis recipes, which consist of precursors, temperatures, and operational sequences, vary in descriptions due to personal preference, creating a significant bottleneck for reproducibility and new material discovery.

Constructing a material synthesis knowledge graph offers a solution by systematically mapping relationships between target materials, precursors, and thermodynamic conditions. This enables predictive modeling and inverse design, bridging the gap between computational materials design and practical laboratory synthesis.

For material synthesis text, the schema and relation are especially important. Thus, the task is not simply entity recognition but primarily relation and event extraction. Generally, for such a specific domain, a traditional fine-tuned information extraction model could perform really well. However, the key problem is the lack of resources, as there is almost no annotated synthesis dataset in materials synthesis. LLMs are powerful at reducing costs, both in time and effort. Bai et al. introduced an autonomous knowledge graph construction pipeline using an LLM to define the schema and extract entities and events from unstructured text (Bai et al., 2025). However, these existing approaches rely heavily on LLM performance, which may not align with the desired schema. In addition, errors propagate in the graph if they exist, making manual curation necessary. To address these issues, we present VKGC, a three-stage knowledge graph construction framework via 1) training a light fine-tuned domain-specific model, 2) correcting the result of the light fine-tuned model by an LLM, and 3) manually curating the result via domain experts to ensure correctness. We used this framework to construct a verifiable material synthesis planning knowledge graph and a QA dataset to benchmark an LLM's knowledge of material synthesis.

Ultimately, the goal of a material synthesis knowledge graph is to enable predictive modeling and inverse design. With a comprehensive machine-readable map of historical synthesis data, researchers can train advanced machine learning models, such as diffusion models and Graph Neural Networks, to recommend optimal synthesis routes for entirely new, hypothetical materials (Pan et al., 2025; Reiser et al., 2022). This could bridge the critical gap between computational materials design, which predicts a material's properties, and the practical material synthesis of actually creating it in the lab. By reducing reliance on expensive, time-consuming experimental trials, a material synthesis knowledge graph serves as foundational infrastructure for the next generation of automated, AI-driven laboratories.

1.3 Contributions

In summary, the key contributions of this paper are as follows:

- We formulate material synthesis information extraction as a low-resource, schema-sensitive scientific information extraction task involving entities, relations, and procedural events.
- We propose an incremental hybrid framework, VKGC, that combines domain-tuned generative information extraction, LLM-based correction, and expert verification to construct a verifiable knowledge graph.
- We build an expert-grounded synthesis knowledge graph and benchmark for synthesis route planning and show that frontier LLMs still fail dramatically on this task despite high confidence.
- We provide a comprehensive analysis of data efficiency, LLM correction gains, schema alignment, and model failure in scientific information extraction.

2 Related Work

2.1 Information extraction via Generation

Information Extraction (IE) is a fundamental natural language processing task that includes Named Entity Recognition (NER) as well as Relations Extraction (RE) and Event Extraction (EE) from unstructured text (Wilks, 1997). IE has traditionally been dominated by extractive architectures, where models such as BERT and RoBERTa are employed to pinpoint specific text spans for tasks (Devlin et al., 2018; Su et al., 2022b). For instance, classical NER relies heavily on sequence-tagging strategies, whereas more complex scenarios, such as nested or discontinuous NER, use span-based paradigms to assign entity types to sub-sequences (Li et al., 2021; Su et al., 2022a). To alleviate the need for complex, dedicated architectures, recent research has shifted toward generative IE. This approach reframes the extraction process as the generation of a structured text sequence, enabling the use of pre-trained language models (PLMs).

A fundamental challenge in generative IE is the misalignment between the structured output space of IE tasks and the natural language output space of generative models. To bridge this gap, researchers have widely adopted output format augmentation,

which converts IE targets into structured text sequences compatible with generative PLMs (Joshi et al., 2020). These formats range from natural-language augmented sequences to special-token sequences (Paolini et al., 2021; Iovine et al., 2022). Furthermore, task transformation has proven effective in stimulating the generalization capabilities of PLMs. Sample methods include reformulating IE as a Question Answering (QA) task, where specific questions are designed to extract corresponding elements, or as a template-filling problem, where generative models fill entities and relations within predefined templates (Kondragunta et al., 2023; Cui et al., 2021).

Finally, the decoding phase in generative IE has seen significant innovations to ensure the validity of the generated structures and mitigate inherent biases. Standard auto-regressive decoding can sometimes produce tokens outside the predefined schema. To prevent this, constrained decoding mechanisms have been introduced to restrict the probability distribution of generated tokens. Dynamic constraints utilize signals such as predicted BIO tags or special tokens to dynamically adjust the vocabulary distribution, whereas static constraints often employ copy mechanisms via pointer networks to extract tokens directly from a fixed scope (Cao and Ananiadou, 2021; Yan et al., 2021). Additionally, to address the order bias introduced by concatenating unordered IE objects into a sequential format-argumented text, set-decoding paradigms have been developed. These approaches utilize query vectors to generate elements of the IE target set in parallel, which optimizes set probability directly and mitigates error propagation (Sui et al., 2020).

2.2 Large Language Model (LLM)

NER and IE have experienced a paradigm shift towards generative frameworks driven by Large Language Models (LLMs), which used to be formulated as a sequence labeling task. To bridge the gap between sequence labeling and text generation, researchers have developed various prompt engineering strategies. For instance, ChatIE transforms zero-shot NER into a multi-turn Question-Answering (QA) problem, while PromptNER utilizes Chain-of-Thought (CoT) prompting alongside prompt-based heuristics and entity definitions to guide the model's reasoning (Wei et al., 2023; Ashok and Lipton, 2023). To mitigate the hallucination issues inherent in LLMs, recent work has introduced self-verification and self-improvement mechanisms. GPT-NER employs a self-verification strategy to address the mislabeling of NULL inputs, and Xie et al. proposed a training-free framework that utilizes LLMs to generate pseudo-demonstrations on unlabeled corpora for iterative self-improvement (Wang et al., 2023; Xie et al., 2024).

Although LLMs exhibit strong zero-shot and few-shot capabilities, Supervised Fine-Tuning (SFT) and data augmentation remain highly effective for optimizing NER performance. LLMs are increasingly utilized as data annotators and synthetic data generators to overcome resource limitations. For example, LLMaAA employs LLMs as annotators within an active learning loop, and ProgGen generates entity terms and the surrounding context to handle intricate structures (Zhang et al., 2023; Heng et al., 2024).

LLMs often rely on Retrieval-Augmented Generation (RAG) to process extensive unstructured texts. Traditional RAG methodologies that rely on dense chunk embeddings often struggle with efficiency and context relevance. To address these limitations, Lopez et al. (2025) proposed CLinical Entity Augmented Retrieval (CLEAR), an entity-centric RAG pipeline designed to optimize clinical information extraction (Lopez et al., 2025). Instead of using arbitrary embedded text chunks, CLEAR uses zero-shot NER, coupled with ontology and LLM-based augmentation, to identify and retrieve highly relevant context windows surrounding specific target entities. They evaluated multiple LLMs and demonstrated that this entity-augmented approach not only improves extraction accuracy, but also yields substantial computational efficiencies.

3 Incremental Knowledge Graph-driven Verifiable Materials Synthesis

3.1 Overview

(i) The information extraction procedure involves training a LongT5 Model using the TANL Sequence-to-Sequence paradigm. The output is corrected by a Large Language Model, and the result is manually curated to ensure accuracy. Then the curated result is added to the training corpus to retrain the LongT5 model. (ii) knowledge graph Construction part involves the construction of a subgraph using manually curated data. The graph is verified via an automated graph verifier and manual curation in case there are any major errors in the subgraph. An example of a major error is missing

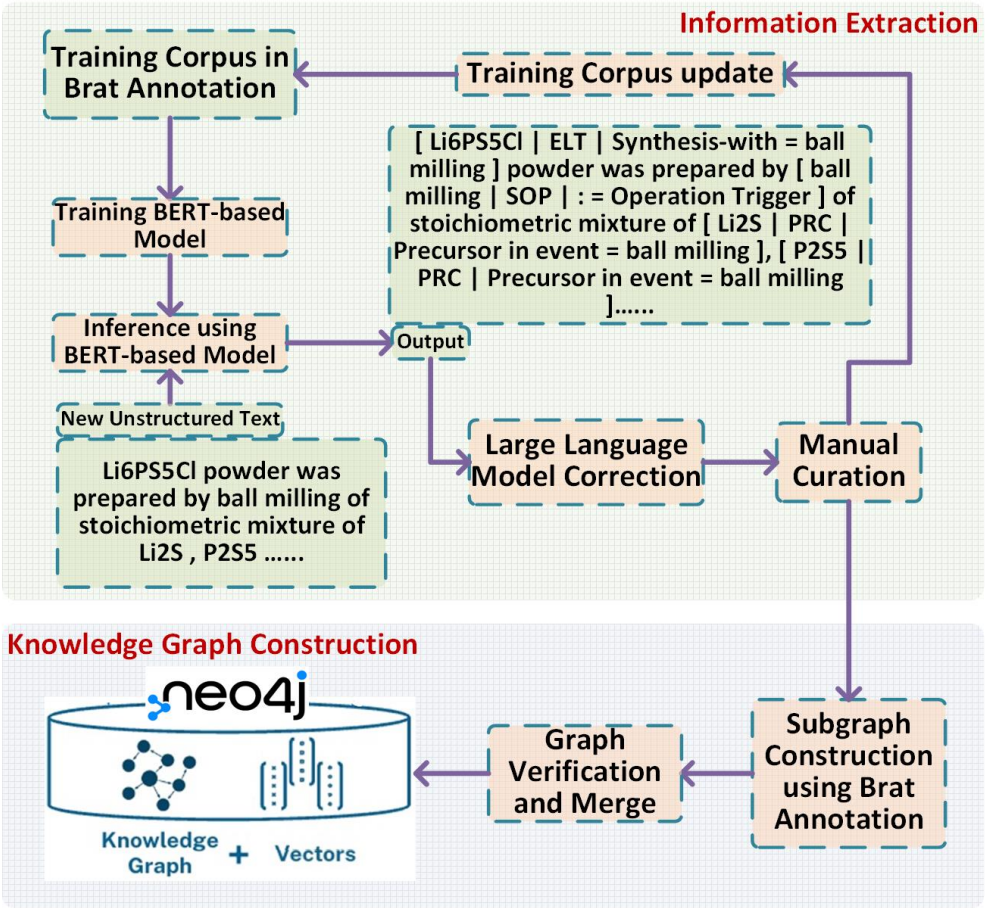

Figure 1: The Overall framework of the VKGC framework that consists of (i) Information Extraction loop and (ii) Knowledge Graph Construction.

edges, which is usually due to a missing relation. The auto verifier can detect missing edges, but is hardly able to identify which nodes to connect, which thus need manual curation. The verified subgraph is merged into the large database in the Neo4j graph database.

3.2 Data Collection and Annotation

We designed Labels for Entity, Relation, and Event for Material Synthesis. The goal is to construct a small knowledge subgraph for each experiment using extracted information, which will later be merged into a complete knowledge graph. This is why we decided to include event extraction, which can capture relational information in a more structured manner. We defined operation words, such as stir, as triggers, and then defined entities, such as precursors, as event arguments. As mentioned before, the goal of the knowledge graph is to enable machine learning-based methods, such as GNNs, to learn hidden patterns in material synthesis and, ultimately, to support lab and experiment automation.

3.3 Sequence-to-Sequence based Information Extraction

We primarily followed the work of TANL: Structured Prediction as Translation between Augmented Natural Languages, but we modified their approach on the event extraction part. Before explaining the modification, it is necessary to explain in detail what they did in the original work. TANL formats the information extraction as text translation. Specifically, the model takes a raw natural language sentence as input and generates a new sequence that annotates the input sentence with specific structural symbols (brackets [ ] and pipes |). For example, they turn the input sentence "Tolkien's epic novel The Lord of the Rings was published in 1954-1955, years after the book was completed." into "[Tolkien | person]'s epic novel [The Lord of the Rings | book | author = Tolkien] was published in 1954-1955, years after the book was completed." where they can parse the output

sentence into actual entity spans and relations. Besides, instead of using abbreviations (e.g., PER or ORG), the model uses natural language words (e.g., person, organization).

In the original work, event extraction involves extracting the event trigger from the text first. Then, the trigger and text have to go through the model for event argument extraction. In addition, they pass one trigger at a time, which increases the time required when there are multiple triggers. We slightly modified this approach by introducing a new special token ":". For trigger extraction, if an entity is a trigger word, we formulate output in the natural language form as "[Entity | := {Trigger Type} Trigger]". Then, for the event argument, we formulate output in the natural language form as "[Entity | {Event Argument Type} in Event = {Trigger}]", which is essentially a special type of relation.

3.4 Entity, Relation, and Event Extraction

To construct the knowledge for the material synthesis route, the first step is to extract information from unstructured scholarly papers in Materials Science. We adopted a method that uses LongT5 and an LLM to generate annotations in Brat format. We first extracted the experimental sections from a curated list of influential papers using an LLM. After performing initial manual annotations, we trained a LongT5 Model in a sequence-to-sequence paradigm.

We used the initially trained LongT5 Model to generate annotations for the remaining extracted experiments. Then, we fed the extracted results and the original text to a frontier LLM for correction. We carefully reviewed the LLM-corrected annotations and manually corrected them to ensure all labels were correct. In addition to the model's performance, we conducted experiments on generation stability (such as early or missing End Of Sequence tokens) and evaluated the contribution of the LLM correction in information extraction. Full dataset statistics, model versions, and details of how generation failure analysis is performed are included in Appendix E.2.

3.5 Incremental Benchmark Construction

Using the information we extracted from the papers, we used a combination of LongT5, LLM, and manual curation to construct an expert-level Knowledge graph using Neo4j. We first construct subgraphs for each experiment using the manually curated Brat annotation. Then, we verify if the constructed subgraph is complete and connected. If any nodes are missing, such as the purity of the precursor, we use the most common purity value and automatically complete the graph. Errors that cannot be resolved automatically are manually corrected.

After all data were stored in the Neo4j database, we built a QA benchmark dataset for material synthesis route planning. We prompted an LLM to generate synthesis routes for sampled electrolytes using relevant information retrieved from the knowledge graph. To evaluate frontier models on this benchmark, we use an LLM as a judge to evaluate if the generated synthesis paths match the answers generated using retrieved information in the QA dataset. We also measured average confidence scores to assess model calibration. Specific details regarding the sampling, models used, and evaluation metrics are provided in Appendix E.3.

4 Experimental Results and Discussion

4.1 Can a small amount of domain annotation support high-quality scientific IE?

We systematically evaluated the performance of our model in information extraction, including both NER, RE, and EE. As mentioned, for event extraction, we only evaluate the event argument. If the predicted event tuple matches the ground truth, the trigger should automatically be correct. As shown in Table 1, the model's performance degrades significantly when the dataset size drops below 60. The degradation is mainly due to recall. The entity precision and event argument precision stay high across all experiments. The precision of the relation drops slightly once the size is below 60. Our UIE baseline performs poorly in comparison, which we assume is caused by nested events, which we need for a clear, structured knowledge graph. The goal is to maintain the nested event, mainly the Condition, at the same level as others, such as Precursors.

In comparison, our fine-tuned LongT5 model shows surprising performance. The assumption is that the material synthesis procedures are highly similar across papers. Precursors and participants often take the form of chemical expressions, making them easy to identify. Besides, we prompted the LLM to extract the experiment section, which increases the similarity as the model inevitably paraphrases or summarizes parts of the text. Compared to datasets like ACE2005, our dataset is far

Dataset Size	Entity			Relation			Event Argument		
	F1	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision
40	93.72	88.90	99.10	89.95	84.25	96.46	91.11	86.13	96.69
60	92.27	86.01	99.51	86.14	80.15	93.08	88.49	81.08	97.40
80	98.95	98.09	99.84	**97.91**	97.11	**98.74**	94.77	91.27	98.55
100	98.96	98.09	99.85	97.69	97.05	98.35	97.11	94.74	**99.59**
136	**99.11**	**98.37**	**99.86**	97.84	**97.29**	98.41	**97.18**	**95.19**	99.25
Baselines									
UIE	37.49	26.89	61.84	29.19	20.35	51.61	-	-	-
Deepseekv3.2	49.13	51.61	46.88	21.15	24.68	18.50	5.89	5.92	5.85
Grok 4.20	48.47	42.49	56.42	22.08	19.75	25.03	2.73	2.12	3.84
Qwen 3.5 Flash	69.79	68.86	70.74	37.41	35.15	39.97	6.95	6.04	8.17
Qwen 3.5 397b-a17b	**75.67**	**77.20**	**74.20**	**51.75**	50.23	**53.36**	**31.40**	32.70	**30.19**
GLM 5.1	69.63	75.46	64.64	45.76	51.15	41.39	26.89	**32.81**	22.78
Gemini 3 Flash	71.20	78.29	65.29	49.57	**54.68**	45.34	28.85	32.10	26.21

Table 1: LongT5 Model performance using different dataset sizes. As the dataset size grows, performance improves, and with over 60 training sizes, the performance is good enough for inference. The bottom part of the table compares the top LLMs' performance. The entity extraction part is relatively good, and the event arguments extraction part is generally poor. All three extraction performances fall far behind the task-specific LongT5 Model.

less diverse, which makes it easier to learn. Some domain-specific text, such as medical records, may also contain highly similar text, and thus information extraction on such text may perform as well as the result shown in our domain. Interestingly, precision is relatively high across all three information types, and recall is the one that lowers the F1 score. This is an upside of manual curation, since it means experts would spend less time on already-extracted information, reducing the labor required in the most costly yet necessary step.

Dataset Size	Failed Sample Count	Total Sample Count
40	34	141
60	35	141
80	17	141
100	14	141
136	**9**	141

Table 2: This table shows the failed generations. We manually counted the number of failed generations in which the model failed to output the EOS Token, or output it too early, causing it to keep producing repeated or incomplete sentences.

We also manually counted the number of failed generations for all 5 models trained on datasets of different sizes. As shown in Table 2, the size of the dataset increases as the number of failed generations decreases. Similar to the result in the Table 1, once the dataset size exceeds 60, the failure rate almost cuts in half. In addition, we manually evaluated the failed generations. Although it fails to generate the EOS Token, it only outputs duplicates of the last sentence it produced, which can be automatically removed using set operations. Once the output sentences are parsed into the Brat annotation format, the only difference is that the annotations are partially missing, starting at the position of the repetition. Other extracted entities, relations, and event arguments will not be influenced. It is the same for the early EOS Token, where the model stops before annotating all text, and the already-annotated information remains unaffected. In practice, a failed generation means that you need to do more manual annotations after the repetition or halt, but it will not mess up the whole pipeline.

4.2 What does LLM correction contribute beyond a domain-tuned extractor and LLM as a extractor?

We evaluated the performance of the LLM Correction. As shown in Table 3, the best model needs the least LLM correction. The results show that the LLM correction helps significantly, as it extracts almost twice as many relations, even with the best LongT5 model trained on the largest dataset. There are also significant increases in the number of entity and event argument extractions, which underscore the importance of the LLM correction procedure.

Dataset Size	Entity Increase	Relation Increase	Event Args Increase
40	93.38%	231.91%	118.42%
60	95.90%	**170.45%**	136.36%
80	93.70%	213.45%	103.36%
100	91.12%	284.85%	122.58%
136	**51.61%**	194%	**56.5%**

Table 3: LongT5 Model Performance Compared to LongT5 Model + LLM. This table shows the percentage increase after LLM correction for Entity, Relation, and Event. A lower score means the LongT5 Model managed to extract more information, and thus the correction generated by the LLM is smaller.

The increased entity, relation, and event arguments are not all correct, as shown in Table 1. LLM managed to extract more information; although it may not be fully correct, it reduces annotation time, since deleting an annotated label is easier than annotating a new one.

We also conducted a case study comparing using an LLM-only approach and an LLM + LongT5 Model approach. The evaluation is qualitative, as we manually compare the results. One sample case is attached in Appendix G. In short, using an LLM-only approach enables the extraction of more entities, relations, and events. However, that extracted information is less aligned with the predefined rules. Using LLM + LongT5 has advantages in defining the type of Entity, Relation, and Event Arguments. Specifically, we defined gas as a Device rather than Participants. Defining gases as participants might be more aligned with common sense. However, since the Gas is usually a limiting factor in material synthesis and is often controlled by devices such as a glovebox, we define it as a Device. A common Large Language Model would often annotate it as a Participant even when the prompt is clear. The LLM-only approach could not consistently extract the label aligned with our preference, whereas the LLM + LongT5 Model approach successfully labeled all gases as Devices. In addition, LLM sometimes failed in annotating a unit separately despite a clear prompt. Our assumption is that using an LLM + LongT5 would give the LLM hints about how the user defines entity types. The information already extracted serves as in-context examples in the prompt, making the generated annotations more closely aligned with the user's preferences.

4.3 Do current frontier LLMs actually possess synthesis route knowledge?

We also generated a QA dataset using the knowledge graph we constructed and evaluated the performance of some recent LLMs. It contains 95 Question-Answer Pairs, and the correct answers may contain more than one synthesis path. From the Table 4, it is clear that current LLMs have few abilities in Material synthesis planning. Besides, most models are confident in their results, with average scores above 60%. It indicates that the models not only lack knowledge but also hold incorrect knowledge and exhibit significant hallucination in material synthesis. LLMs using RAG perform significantly better, but their accuracy remains mostly below 30%, indicating that LLMs struggle to understand experimental relations. State-of-the-art models barely have any prior knowledge in the field. Thus, to address this gap, we provided this QA dataset. This QA dataset will also grow as more and more experiments are extracted, inferred, annotated, and merged into the knowledge graph.

This QA dataset also potentially enables material inverse design. Typically, Materials Scientists want a new material with specific properties. However, inverse material design is complex, and even when scientists know which material they want to synthesize, they may be unable to do so because they lack the procedures. Using the QA dataset, it is possible to train an LLM or a smaller, domain-specific GNN model to predict the synthesis route. Then, combined with lab automation, it is possible to build a fully automated lab with an auto-predicted synthesis route, auto-execution of the route, and auto-collection of results. Besides, the predicted route and its results can be added to the knowledge graph, making the whole framework a closed loop.

Model Name	Plain Prompt Accuracy	Confidence Score	RAG Accuracy	Confidence Score
Deepseekv3.2	1.6%	74.11%	16.84%	84.38%
Grok 4.20	0%	68.46%	23.15%	78.44%
Qwen 3.5 Flash	0%	81.97%	**32.63%**	96.55%
Qwen 3.5 397b-a17b	**4.06%**	78.69%	27.36%	77.33%
GLM 5.1	3.25%	83.36%	24.21%	97.96%
Gemini 3 Flash	**4.06%**	91.42%	18.94%	95.08%

Table 4: This table shows the performance of the LLMs in material synthesis route planning using prompting only.

5 Conclusion

In this work, we propose VKGC, a novel framework designed to address the issues of extracting complex material synthesis procedures from unstructured scientific literature. To address the limitations of relying solely on LLMs or traditional extraction methods, we proposed an incremental, three-stage methodology for constructing a verifiable knowledge graph from unstructured text. Our approach includes: 1) training a light fine-tuned domain-specific model to extract information from unstructured scholarly text, 2) correcting the result annotation of the light fine-tuned model by an LLM, and 3) manually curating the annotations via crowdsourcing experts to ensure correctness. We adopted an LLM + LongT5 approach to extract information from unstructured text, and manually curated extracted information by domain experts to ensure correctness. Human intervention here is crucial and largely irreplaceable, since there is very little data in the domain, and synthesizing data or relying on LLMs' prior knowledge will be impossible. This also applies to any domain with little data currently available.

Our experiments demonstrate that while the fine-tuned LongT5 model achieves surprisingly high precision in identifying entities, relations, and event arguments, its recall degrades in low-resource settings. The integration of LLM-based correction significantly mitigates this limitation by extracting entities, event arguments, and especially more relations, and reducing the labor-intensive burden of manual curation. Importantly, we found that the combined LongT5 and LLM approach produces annotations that are far more aligned with predefined domain rules than the LLM-only approach.

Furthermore, we used the constructed expert-level knowledge graph to develop a QA benchmark dataset tailored for material synthesis route planning. Our evaluation of recent state-of-the-art LLMs on this benchmark revealed a stark deficiency in their ability to generate accurate synthesis routes, underscoring the necessity of our verifiable knowledge graph and dataset. Ultimately, the VKGC framework provides a foundational infrastructure for the future of materials science. By systematically transforming fragmented, unstructured text into machine-readable commands, this work paves the way for advanced predictive modeling and inverse material design. As the knowledge graph continues to grow through incremental extraction and expert verification, it will serve as a critical support in the construction of fully automated, AI-driven laboratories, significantly accelerating the discovery and synthesis of novel materials.

6 Limitation

While the VKGC framework significantly reduces the burden of building domain-specific knowledge graphs, several limitations remain. First, despite the efficiency gains from LLM-based corrections, our pipeline still inevitably relies on human-in-the-loop expert curation to guarantee 100% verifiable accuracy.

Second, our current dataset and benchmark are relatively narrow in scope, focusing primarily on the synthesis of electrolyte materials. While the framework is designed to be domain-agnostic, its performance on other branches of chemistry or entirely different scientific disciplines with more complex nested event structures remains to be evaluated. Besides, our current dataset and benchmark are relatively small, primarily due to the lack of labor. All annotations are done by the authors since the budget is limited and hiring additional domain experts is unaffordable.

Finally, the underlying LongT5 model occasionally suffers from generation instability, such as premature or missing End-Of-Sequence (EOS) tokens. Although it is ignorable in production, future work could explore more robust sequence-to-sequence architectures to eliminate this generation instability

entirely.

AI Assistant Disclosure

We used an AI Assistant to proofread and improve the fluency of minor sections of this manuscript.

License

All artifacts (models, datasets, and software) used in this research were accessed and utilized in strict compliance with their respective licenses and terms of use.

Open-Weights and Open-Source Models:

- LongT5 and UIE are publicly available and utilized under the Apache License 2.0, which permits free use, modification, and distribution for research purposes.

- Qwen 3.5, Deepseek, and GLM models were used in accordance with their respective open-weights research licenses (e.g., the Tongyi Qianwen License and GLM License), which permit academic research and restricted commercial use(Bai et al., 2023; DeepSeek-AI, 2024; Zeng et al., 2022).

Proprietary APIs:

- Gemini 3 (Pro/Flash) and Grok 4.20 were accessed via their respective official APIs(Gemini Team et al., 2023; xAI, 2024). Their usage strictly adheres to the Google API Terms of Service and xAI Terms of Service, respectively. We ensured that no personally identifiable information (PII) or sensitive data was transmitted to these APIs, as our domain strictly focuses on materials science literature.

Software Tools:

- Neo4j Community Edition is licensed under the GPLv3, which is fully compatible with our academic research use case(Neo4j, Inc., 2024).

- The Brat annotation tool is open-source and distributed under the MIT License(Stenetorp et al., 2012).

References

Dhananjay Ashok and Zachary C. Lipton. 2023. Promptner: Prompting for named entity recognition. *Preprint*, arXiv:2305.15444.

Jiaxin Bai, Wei Fan, Qi Hu, Qing Zong, Chunyang Li, Hong Ting Tsang, Hongyu Luo, Yauwai Yim, Haoyu Huang, Xiao Zhou, Feng Qin, Tianshi Zheng, Xi Peng, Xin Yao, Huiwen Yang, Leijie Wu, Yi Ji, Gong Zhang, Renhai Chen, and Yangqiu Song. 2025. Autoschemakg: Autonomous knowledge graph construction through dynamic schema induction from web-scale corpora. *Preprint*, arXiv:2505.23628.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Jiarun Cao and Sophia Ananiadou. 2021. GenerativeRE: Incorporating a novel copy mechanism and pretrained model for joint entity and relation extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2119–2126, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using BART. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1835–1845, Online. Association for Computational Linguistics.

DeepSeek-AI. 2024. Deepseek-llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, and 1 others. 2023. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. 2021. Longt5: Efficient text-to-text transformer for long sequences. *Preprint*, arXiv:2112.07916.

Yuzhao Heng, Chunyuan Deng, Yitong Li, Yue Yu, Yinghao Li, Rongzhi Zhang, and Chao Zhang. 2024. ProgGen: Generating named entity recognition datasets step-by-step with self-reflexive large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15992–16030, Bangkok, Thailand. Association for Computational Linguistics.

Andrea Iovine, Anjie Fang, Besnik Fetahu, Oleg Rokhlenko, and Shervin Malmasi. 2022. Cyclener:

An unsupervised training approach for named entity recognition. pages 2916–2924.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the association for computational linguistics*, 8:64–77.

Murali Kondragunta, Olatz Perez-de Viñaspre, and Maite Oronoz. 2023. Improving and simplifying template-based named entity recognition. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 79–86, Dubrovnik, Croatia. Association for Computational Linguistics.

Fei Li, ZhiChao Lin, Meishan Zhang, and Donghong Ji. 2021. A span-based model for joint overlapped and discontinuous named entity recognition. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4814–4828, Online. Association for Computational Linguistics.

I. Lopez, A. Swaminathan, K. Vedula, S. Narayanan, F. Nateghi Haredasht, S. P. Ma, A. S. Liang, S. Tate, M. Maddali, R. J. Gallo, N. H. Shah, and J. H. Chen. 2025. Clinical entity augmented retrieval for clinical information extraction. *npj Digital Medicine*, 8(1):45.

Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. 2022. Unified structure generation for universal information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5755–5772, Dublin, Ireland. Association for Computational Linguistics.

Neo4j, Inc. 2024. Neo4j graph database platform. `https://neo4j.com/`.

Elton Pan, Soonhyoung Kwon, Sulin Liu, Mingrou Xie, Alexander J. Hoffman, Yifei Duan, Thorben Prein, Killian Sheriff, Yuriy Roman-Leshkov, Manuel Moliner, Rafael Gomez-Bombarelli, and Elsa Olivetti. 2025. Diffsyn: A generative diffusion approach to materials synthesis planning. *Preprint*, arXiv:2509.17094.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. 2021. Structured Prediction as Translation between Augmented Natural Languages. In *International Conference on Learning Representations (ICLR)*.

P. Reiser, M. Neubert, A. Eberhard, L. Torresi, C. Zhou, C. Shao, H. Metni, C. Van Hoesel, H. Schopmans, T. Sommer, and P. Friederich. 2022. Graph neural networks for materials science and chemistry. *Communications Materials*, 3(1):93.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107. Association for Computational Linguistics.

Jianlin Su, Ahmed Murtadha, Shengfeng Pan, Jing Hou, Jun Sun, Wanwei Huang, Bo Wen, and Yunfeng Liu. 2022a. Global pointer: Novel efficient span-based approach for named entity recognition. *Preprint*, arXiv:2208.03054.

Ming-Hsiang Su, Chin-Wei Lee, Chi-Lun Hsu, and Ruei-Cyuan Su. 2022b. RoBERTa-based traditional chinese medicine named entity recognition model. In *Proceedings of the 34th Conference on Computational Linguistics and Speech Processing (ROCLING 2022)*, pages 61–66, Taipei, Taiwan. The Association for Computational Linguistics and Chinese Language Processing (ACLCLP).

Dianbo Sui, Yubo Chen, Kang Liu, Jun Zhao, Xiangrong Zeng, and Shengping Liu. 2020. Joint entity and relation extraction with set prediction networks. *Preprint*, arXiv:2011.01675.

Shuhe Wang, Xiaofei Sun, Xiaoya Li, Rongbin Ouyang, Fei Wu, Tianwei Zhang, Jiwei Li, and Guoyin Wang. 2023. GPT-NER: Named entity recognition via large language models. *arXiv preprint arXiv:2304.10428*.

X. Wei, X. Cui, N. Cheng, X. Wang, X. Zhang, S. Huang, P. Xie, J. Xu, Y. Chen, M. Zhang, Y. Jiang, and W. Han. 2023. Chatie: Zero-shot information extraction via chatting with chatgpt. *Preprint*, arXiv:2302.10205.

Y. Wilks. 1997. Information extraction as a core language technology. In M. T. Pazienza, editor, *Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology*, volume 1299 of *Lecture Notes in Computer Science*, pages 1–9. Springer, Berlin, Heidelberg.

xAI. 2024. Grok. `https://x.ai/`.

Tingyu Xie, Qi Li, Yan Zhang, Zuozhu Liu, and Hongwei Wang. 2024. Self-improving for zero-shot named entity recognition with large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 583–593, Mexico City, Mexico. Association for Computational Linguistics.

Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021. A unified generative framework for various NER subtasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5808–5822, Online. Association for Computational Linguistics.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, and 1 others. 2022. Glm-130b: An open bilingual pre-trained model. In *The Eleventh International Conference on Learning Representations*.

Ruoyu Zhang, Yanzeng Li, Yongliang Ma, Ming Zhou, and Lei Zou. 2023. LLMaAA: Making large language models as active annotators. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13088–13103, Singapore. Association for Computational Linguistics.

A Definition of Entity, Relation, Events

Entities:

ELT(Electrolyte)
PRC(Precursor)
DEV(Device)
PTP(Participant)
OPT(OperationTarget)
NUM(Number)
NUT(NumberUnit)
PMC(PropertyMisc)
Event:

Trigger: SOP(SynthesisOperation), Arguments: PRC(Precursor), DEV(Device), PTP(Participant), OPT(OperationTarget), CND(Condition)

Trigger: CND(Condition), Arguments: DEV(Device), PTP(Participant), NUM(Number)

Relations: Electrolyte—>SynthesisWith—>SynthesisOperation

Precursor/Participant/OperationTarget/Electrolyte—>InstanceOf—>Precursor/Participant/OperationTarget/Electrolyte

PropertyMisc—>PropertyOf—>Precursor/Participant/OperationTarget/Device

Number—>AmountOf—>Device

Number—>AttributeOf—>Precursor

Number—>ValueOf—>PropertyMisc

Unit—>UnitOf—>Number

Number—>NumberOf—>Precursor/Participant/OperationTarget

SynthesisOperation—>NextOperation—>SynthesisOperation

B Experiment Extraction Prompt for LLM

You are an expert in Chemistry. Your job is to extract only the synthesis procedures of electrolytes from the following text. Pay attention to the experimental section or the material method section, if exists.

Ignore the Discussion and Result sections if any electrolytes are mentioned in those sections. Be concise, precise, and complete. Make sure to include all numerical details such as temperature, time, (brand and purity of the precursors), and etc.

Do not list the procedure. Instead, use the same tone as the original text and extract the procedure in a paragraph. Use the original text when appropriate.

Avoid overly simplified sentences such as "heating at 350°C for 3 hours and 400°C for 1 hour"; use "heating at 350°C for 3 hours and then heating at 400°C for 1 hour" instead.

Focus exclusively on the preparation of electrolyte materials.

Transform any LaTeX-based chemical expression into a plain string-based chemical expression. Do not include any information related to the preparation of cathode or anode materials, cell assembly, the testing of the materials or the cells, or the proposed usage of the materials. Be aware, one paper could contain more than one experiment. Thus, format your output in a list where each paragraph contains exactly one experiment. If multiple electrolytes are synthesized, and the procedure is abbreviated, expands it.

Example:

Text: "LSPS is synthesized via heating material at 350°C for 3 hours, and for Li6PCl, 300°C for 2 hours."

Output: ["LSPS is synthesized via heating material at 350°C for 3 hours", "Li6PCl is synthesized via heating material at 300°C for 2 hours."]

Final Output Example: ["LSPS is synthesized via ...", "Li6PCl is synthesized via ..."]

Here is the text: text"""

C Correction Prompt for LLM

You are an expert in Chemistry, your job is to extract Entities, Events, and Relations from text and output in JSON Format:

Here are allowed entities, it will be in format [Words, Number of occurrence in the paragraph (i.e. 1 means the first appeared words), entity type, Entity ID (i.e. entity 1)]: Electrolyte, Precursor, Device (label any gas as Device), Participant, OperationTarget (product of the current operation), Number, Unit (for purity with percent %, do not label it as unit but include the percent symbol as Number), PropertyMisc

Example: ['Single-crystalline NCM811', 1, 'Precursor', 'entity 1']

Here are allowed events, it will be in format

[Words, Number of occurrence in the paragraph (i.e. 1 means the first appeared words), type of the event trigger, Event Arguments (use Entity Type + entity ids or Event trigger type + event ids here), Event ID (i.e. event 1)]:

Event Trigger Type: SynthesisOperation, Event Arguments: Precursor, Participant, OperationTarget, Condition (i.e. operation may need specific temperature)

Event Trigger Type: Condition, Event Arguments: Device, Participant, Number

Example: ['synthesized', 1, 'SynthesisOperation', ['Precursor', 'entity 10', 'Condition', 'event 2'], 'event 1']

Do not contain any non-allowed Event Arguments.

Here are allowed relations, it will be in format entity/event–relation–entity/event to specify allowed relations between specific entity types, if multiple entity types are allowed, they will be separated by / symbol:

Electrolyte–SynthesisWith–SynthesisOperation

Precursor/Participant/OperationTarget/Electrolyte–InstanceOf–Precursor/Participant/OperationTarget/Electrolyte

PropertyMisc–PropertyOf–Precursor/Participant/OperationTarget/Device

Number–AmountOf–Device

Number–AttributeOf–Precursor

Number–ValueOf–PropertyMisc

Unit–UnitOf–Number

Number–NumberOf–Precursor/Participant/OperationTarget

SynthesisOperation–NextOperation–SynthesisOperation

Example: ['entity 3', 'ValueOf', 'entity 20']

Your Final JSON output should use the following keys: Entities, Events, and Relations

There are some already extracted Entities, Events, and Relations. You should correct them by adding Entities, Events, and Relations that are not extracted or revise them.

For already extracted Entities, they will be in the format of: (Words, entity type, Entity ID (i.e. entity 1))

Example:{{('mixed powder', 'OperationTarget', 'entity 16')}}

For already extracted Events, they will be in the format of: (Words, event type, Event ID (i.e. event 1), Event Arguments (in tuple))

Example:{{('2 hours', 'Condition', 'event 5', ('Number', '2', 'Unit', 'hours'))}}

For already extracted Relations, they will be in the format of: (entity/event, relation, entity/event)

Example:{{('event 7', 'NextOperation', 'event 10')}}

Here is the text: {text}

Here is the already extracted Entities: {entity}

Here is the already extracted Events: {event}

Here is the already extracted Relations: {relation}

Now Start your extraction, REMEMBER TO OUTPUT IN JSON

D Entity, Relation, Event Arguments Extraction Prompt for LLM

You are an expert in Chemistry, your job is to extract Entities, Events, and Relations from text and output in JSON Format:

Here are allowed entities, it will be in format [Words, Number of occurrence in the paragraph (i.e. 1 means the first appeared words), entity type, Entity ID (i.e. entity 1)]: Electrolyte, Precursor, Device (label any gas as Device), Participant, OperationTarget (product of the current operation), Number, Unit (for purity with percent %, do not label it as unit but include the percent symbol as Number), PropertyMisc

Example: ['Single-crystalline NCM811', 1, 'Precursor', 'entity 1']

Here are allowed events, it will be in format [Words, Number of occurrence in the paragraph (i.e. 1 means the first appeared words), type of the event trigger, Event Arguments (use Entity Type + entity ids or Event trigger type + event ids here), Event ID (i.e. event 1)]:

Event Trigger Type: SynthesisOperation, Event Arguments: Precursor, Participant, OperationTarget, Condition (i.e. operation may need specific temperature)

Event Trigger Type: Condition, Event Arguments: Device, Participant, Number

Example: ['synthesized', 1, 'SynthesisOperation', ['Precursor', 'entity 10', 'Condition', 'event 2'], 'event 1']

Do not contain any non-allowed Event Arguments.

Here are allowed relations, it will be in format entity/event–relation–entity/event to specify allowed relations between specific entity types, if multiple entity types are allowed, they will be separated by / symbol:

Electrolyte–SynthesisWith–SynthesisOperation

Precursor/Participant/OperationTarget/Electrolyte–InstanceOf–Precursor/Participant/OperationTarget/Electrolyte

PropertyMisc–PropertyOf–Precursor/Participant/OperationTarget/Device

Number–AmountOf–Device

Number–AttributeOf–Precursor

Number–ValueOf–PropertyMisc

Unit–UnitOf–Number

Number–NumberOf–Precursor/Participant/OperationTarget

SynthesisOperation–NextOperation–SynthesisOperation

Example: ['entity 3', 'ValueOf', 'entity 20']

Your Final JSON output should use the following keys: Entities, Events, and Relations

Here is the text: {text}

Now start your extraction, REMEMBER TO OUTPUT IN JSON

E Experiment Detail

E.1 Model Training and Baseline Details

We conducted 5 experiments using all 136 annotations, 100 annotations, 80 annotations, 60 annotations, and 40 annotations to train the LongT5 Model. The base model we used is the LongT5 model, a Model with sliding-window attention and transient global attention for long text sequence-to-sequence tasks (Guo et al., 2021). In all experiments, we trained the model for 50 epochs and used a train-validation-test split of 0.5:0.3:0.2. Training take about 4 hours and 12 minutes for 136 annotations; 3 hours and 48 minutes for 100 annotations; 3 hours and 10 minutes for 80 annotations; 2 hours and 24 minutes for 60 annotations; 59 minutes for 40 annotations on 1 NVIDIA A100 GPU. The results we reported for our model are the average of 3 runs.

Since we extract triggers and arguments simultaneously, the metric we use is event-argument accuracy. For each event argument, the output will be a tuple "(Event Argument Type, (Entity\Event Type, Token Start Index, Token index), (Trigger Type, Trigger Start Token Index, Trigger End Token Index))". This metric can also evaluate the performance of trigger extraction. The trigger must be correct for the whole tuple to be correct. As a comparison, we also present some baseline result in the same table, and the prompt used is provided in the appendix. The baselines we used are the sequence-to-sequence-based method Unified Structure Generation for Universal Information Extraction (UIE) (Lu et al., 2022), which is a strong sequence-to-sequence baseline, and LLM-only extraction. We followed the original work, converted our Brat annotation into their data format, and trained the model using their default settings. We planned to include a span-based extraction method, but span-based methods struggle in our dataset due to nested events. We omit the comparison with the original TANL since the main purpose of our modification is to skip the trigger extraction procedure, and the original TANL needs that step for event extraction.

E.2 Data Collection and Extraction Statistics

To build the dataset, we manually selected 116 relatively influential papers in the field of Materials Science. We used the Qwen3.5-397B-A17B model and fed the entire paper in text form to extract only the experimental parts. We prompted the LLM to extract multiple experiments separately and store them in different files, ensuring each file contains only one experiment. From all 116 papers, we extracted 136 experiments. We initially did 13 manual annotations and trained the LongT5 Model. The full-size Qwen3.5-397b-a17b was subsequently used for correcting the LongT5 outputs.

During generation, the parser used set operations to parse the output sentences, which automatically removes duplicates. However, the trained LongT5 model sometimes failed to output the End Of Sequence (EOS) token and instead produced duplicate sentences or output the EOS token too early. We collected 141 sample texts and manually counted the number of failed generations across all 5 experiments (Table 2). To evaluate the LLM correction, we randomly sampled 20 texts not used in the training sets, ran inference using the 5 models, and fed them to the LLM for correction to calculate the percentage increase in extracted Entities, Relations, and Events (Table 3).

E.3 Benchmark Construction and LLM Evaluation

For the QA dataset construction, a total of 95 electrolytes were sampled from the Neo4j database. We prompted Gemini 3 Pro to generate material synthesis routes for these electrolytes using relevant information extracted from the knowledge graph. Because one material could have more than one synthesis path, we used an LLM as a judge to com-

pare the generated paths against the retrieved paths from the graph database, rather than asking it to directly judge a single generated path (which is prone to hallucination). We also measured average confidence scores by asking the evaluated models to generate a confidence score between 0 and 100 to indicate how certain they are about their results.

F Software Dependencies and Packages

All experiments were implemented in Python 3.10. The core machine learning pipeline was built using PyTorch and accelerated using Hugging Face `accelerate` (v1.0.1) and Triton (v2.2.0). Graph neural network operations and structural data processing were handled via `torch-geometric` (v2.6.1).

For Large Language Model (LLM) orchestration and retrieval pipelines, we utilized the LangChain framework, specifically leveraging `langchain-core` (v0.2.43) and `langchain-community` (v0.2.19). Knowledge graph storage and querying were executed using the `neo4j` Python driver (v5.27.0). External document parsing and data retrieval used `pypdf` (v5.5.0), `beautifulsoup4` (v4.12.3), and the specialized API wrappers `arxiv` (v2.1.3) and `wikipedia` (v1.4.0).

For inference deployment and acceleration, we used `onnxruntime-gpu` (v1.19.2) and `safetensors` (v0.4.5). Model demonstrations and user interfaces were constructed using Gradio (v4.44.1). Hardware acceleration was mapped via CUDA 12 compatible runtimes (`nvidia-cuda-runtime-cu12`). The complete, exact environment manifest (`requirements.txt`) is provided in our supplementary material repository.

G Case Study

Original Text: Li6PS5Cl (LPSC) powder was obtained by ball milling and solid-state sintering methods, where the mixture of Li2S (99.9%, Hunan Energy Frontiers New Materials Technology Co., LTD), P2S5 (99.9%, Hunan Energy Frontiers New Materials Technology Co., LTD), and LiCl (99.9%, Aladdin) was ball-milled at 200 rpm for 6h followed by a heat treatment at 450°C for 8h.

Structured Output: [Li6PS5Cl | ELT | Synthesis-with = ball milling] (LPSC) powder was obtained by solid-state sintering methods, where the [mixture | OPT | Operation Target in event = ball milling] of [Li2S | PRC | Precursor in event = ball milled] ([99.9% | NUM | Attribute-of = Li2S] , Hunan Energy Frontiers New Materials Technology Co., LTD), [P2S5 | PRC | Precursor in event = ball milled] ([99.9% | NUM | Attribute-of = P2S5] , Hunan Energy Frontiers New Materials Technology Co., LTD), and [LiCl | PRC | Precursor in event = ball milled] ([99.9% | NUM | Attribute-of = LiCl] , Aladdin) was [ball-milled | SOP | Next-Operation = heat treatment | : = Operation Trigger] [at | CND | : = Condition Trigger] [200 | NUM | Number in event = at] [rpm | NUT | Unit-of = 200] [for | CND | : = Condition Trigger] [6 | NUM | Number in event = for] [h | NUT | Unit-of = 6] followed by a [heat treatment | SOP | : = Operation Trigger] [at | CND | : = Condition Trigger] [450 | NUM | Number in event = at] [°C | NUT | Unit-of = 450] [for | CND | : = Condition Trigger] [8 | NUM | Number in event = for] [h | NUT | Unit-of = 8]

LLM Only Brat Annotation:

T1 OPT -1 7 Li6PS5Cl

T2 OPT -1 3 LPSC

T3 PRC -1 3 Li2S

T4 NUM 114 119 99.9%

T5 PMC -1 55 Hunan Energy Frontiers New Materials Technology Co., LTD

T6 PRC -1 3 P2S5

T7 NUM 186 191 99.9%

T8 PMC -1 55 Hunan Energy Frontiers New Materials Technology Co., LTD

T9 PRC -1 3 LiCl

T10 NUM 262 267 99.9%

T11 PMC -1 6 Aladdin

T12 NUM 297 300 200

T13 NUT 301 304 rpm

T14 NUM 309 311 6h

T15 NUM -1 4 450°C

T16 NUM 354 356 8h

LLM + BERT Annotation: T1 OPT 0 8 Li6PS5Cl

T2 OPT 10 14 LPSC

T3 OPT 97 104 mixture

T4 PRC 108 112 Li2S

T5 NUM 114 119 99.9%

T6 PTP 121 177 Hunan Energy Frontiers New Materials Technology Co., LTD

T7 PRC 180 184 P2S5

T8 NUM 186 191 99.9%

T9 PTP 193 249 Hunan Energy Frontiers New Materials Technology Co., LTD

T10 PRC 256 260 LiCl

T11 NUM 262 267 99.9%
T12 PTP 269 276 Aladdin
T13 NUM 297 300 200
T14 NUT 301 304 rpm
T15 NUM 309 310 6
T16 NUT 310 311 h
T17 NUM 344 347 450
T18 NUT 347 349 °C
T19 NUM 354 355 8
T20 NUT 355 356 h