MSP-LLM: A Structured Large Language Model Framework for Inorganic Material Synthesis Planning

Heewoong Noh[1], **Gyoung S. Na**[1], **Namkyeong Lee**[1], **Chanyoung Park**[1]*
[1]KAIST
{heewoongnoh, ngs0726, namkyeong96, cy.park}@kaist.ac.kr

Abstract

Material synthesis planning (MSP) remains a fundamental and underexplored bottleneck in AI-driven materials discovery, as it requires not only identifying suitable precursor materials but also designing coherent sequences of synthesis operations to realize a target material. Although several AI-based approaches have been proposed to address isolated subtasks of MSP, a methodology for solving the entire MSP task has yet to be established. We propose MSP-LLM, a structured LLM-based framework that formulates MSP as a two-stage process composed of two constituent subproblems: precursor prediction (PP) and synthesis operation prediction (SOP). Our approach introduces a discrete material class as an intermediate decision variable that organizes both tasks into a chemically consistent decision chain. For SOP, we further incorporate hierarchical precursor types as synthesis-relevant inductive biases and employ precursor constraint factorization that preserves precursor-related information in the autoregressive decoding state. Extensive experiments show that MSP-LLM consistently outperforms existing methods on both PP and SOP, as well as on the MSP task, demonstrating an effective and scalable framework with practical potential for autonomous MSP. The source code for MSP-LLM is available at https://github.com/HeewoongNoh/MSP-LLM.

1 Introduction

Recently, Artificial Intelligence (AI) has achieved notable successes in condensed matter physics and materials science, primarily by enabling a high-throughput screening at the scales of real-world materials spaces [36, 18, 7, 20]. After exploring the materials space, we are then confronted with the next scientific challenge of designing a synthesis plan to realize the discovered materials [33]. However, most existing machine learning methods have primarily focused on exploring the materials space while overlooking the next challenge in **material synthesis planning (MSP)** [16, 34], and this oversight remains a bottleneck of AI for science in real-world materials discovery. Although numerous synthesis planning methods have been developed for organic compounds, benefiting from well-specified molecular structures and well-defined reaction rules, MSP remains a long-standing challenge in AI for science due to two reasons: (1) The lack of atomic structures for the product materials causes multiple synthesis paths on polymorphic structures, thereby substantially increasing uncertainty in MSP [23, 25]. Moreover, (2) fully specifying the synthesis paths based solely on a target material's composition remains a challenge, as compositional data alone fails to provide a comprehensive blueprint for the necessary chemical reactions [24, 28].

In real-world applications, MSP is explored within a materials space defined at the level of chemical compositions [23, 25, 4], and requires predicting a set of precursor materials P and a sequence of synthesis operations O for a given desired product material. Here, $P \in 2^{\mathcal{M}} \setminus \{\emptyset\}$ denotes a set of precursor materials, where $\mathcal{M}$ denotes the set of materials, and $O \in \mathcal{O}^*$ denotes a finite-length

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

synthesis operation sequence. Formally, the MSP task can be defined as $f : \mathcal{M} \rightarrow 2^{\mathcal{M}} \times \mathcal{O}^*$, which maps a given product material to a set of precursor materials and a corresponding synthesis operation sequence. However, due to the inherent challenges of MSP, most existing studies have addressed only partial aspects of MSP, rather than providing a framework that solves MSP in its entirety. In particular, prior works typically focus on one of the following two subproblems: (1) **precursor prediction (PP)**, which predicts a set of precursor materials from a given product material, $f_p : \mathcal{M} \rightarrow 2^{\mathcal{M}}$, or (2) **synthesis operation prediction (SOP)**, which aims to predict a sequence of synthesis operations given a product material and a set of precursor materials, $f_o : \mathcal{M} \times 2^{\mathcal{M}} \rightarrow \mathcal{O}^*$.

Among these subproblems, SOP remains largely underexplored and has been studied only in narrowly scoped settings, typically restricted to specific material systems rather than general MSP [23, 26]. By contrast, most existing efforts have focused on PP, yet these approaches often rely on complex retrieval pipelines [13, 24] or access to proprietary LLM APIs [17, 28], which limits their scalability and reproducibility. These limitations underscore the need for the scalable modeling framework that can address both SOP and PP within a general MSP setting.

Recently, large language models (LLMs) fine-tuned for domain-specific scientific tasks have demonstrated strong capabilities across scientific domains, particularly in materials science, including generation of candidate materials [11] and emulation of scientific discovery processes [6]. This modeling framework is particularly well-suited to MSP, where all relevant input information is naturally represented as text in the form of chemical compositions [28]. By leveraging LLMs that have been pre-trained on large-scale and diverse textual corpora, LLM-based MSP models can potentially access and utilize extensive background knowledge, enabling more informed and physically grounded decisions [3, 31] in both precursor selection and synthesis operation planning.

However, despite their promise, effective strategies for fine-tuning LLMs to address MSP remain largely unexplored. Specifically, no existing work has established a principled fine-tuning strategy that enables LLMs to perform PP and SOP in a structured MSP framework.

In this paper, we propose a structured LLM-based framework, called MSP-LLM, for solving complex MSP tasks. We develop novel task-specific LLM fine-tuning strategies for the two constituent subproblems of MSP, i.e., PP and SOP. Specifically, (i) we introduce a discrete material class variable that captures synthesis-relevant characteristics of materials and employ it in both PP and SOP as an intermediate decision variable. By incorporating this variable, each subtask is organized into a structured decision chain, in which the model conditions its task-specific generation on the material class variable, leading to more accurate predictions. Furthermore, for SOP, we introduce an explicit conditioning strategy that integrates hierarchical precursor types as synthesis-relevant inductive biases and enforces their effective utilization through precursor constraint factorization, promoting precursor-conditioned operation generation and more reliable retention of precursor constraints in the decoder state. By integrating LLMs fine-tuned for each subproblem within a single coherent framework, MSP-LLM performs MSP given target material compositions. Extensive experiments demonstrate that MSP-LLM significantly outperforms existing methods on both individual subproblems and the MSP task, highlighting its effectiveness and practical applicability to real-world materials discovery.

To the best of our knowledge, MSP-LLM is the first work to formulate and tackle both PP and SOP within a single MSP framework, and to show that this can be effectively addressed by fine-tuning LLMs in a structured manner.

In this study, we make the following contributions:

- We propose MSP-LLM , a structured framework for general Material Synthesis Planning (MSP) that formulates the task as a two-stage process consisting of (1) precursor prediction (PP) and (2) synthesis operation prediction (SOP), each implemented using fine-tuned LLMs tailored to the respective subtask.
- By incorporating domain-specific inductive biases via a material group decision chain in both PP and SOP, together with an explicit conditioning strategy, our framework effectively leverages precursor information in SOP. We further analyze the mechanisms underlying the resulting performance gains.
- Through extensive experiments, we demonstrate that the proposed framework significantly outperforms all baselines on the MSP task as well as its subtasks, highlighting the superiority of our approach.

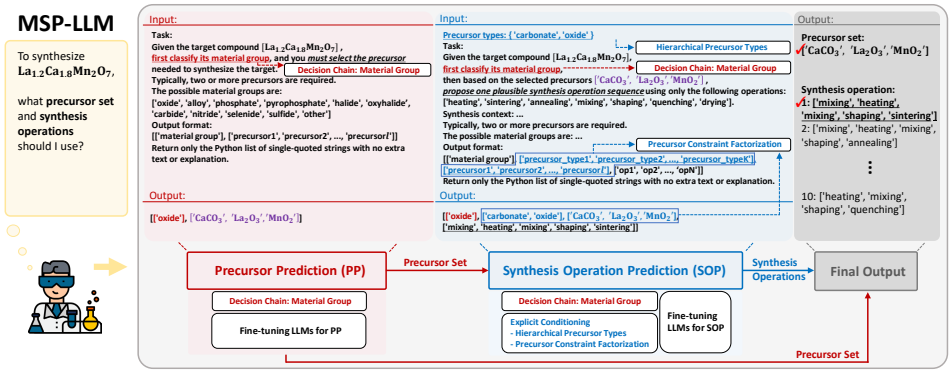


Figure 1: Overall framework of MSP-LLM. Given a target material, MSP-LLM first predicts the precursor set (PP), and then, conditioned on the predicted precursors, generates the synthesis operation sequence (SOP), resulting in a MSP. The input–output texts above the PP and SOP LLMs illustrate example prompts used during LLM fine-tuning for each task.

2 Preliminaries

Material Synthesis Planning (MSP). As introduced in Section 1, MSP associates a target material with a precursor set and a synthesis operation sequence. We adopt a two-stage formulation that emulates the experimental workflow of synthesis chemists who first identify feasible precursors and then design operations conditioned on them. While a one-stage approach can, in principle, learn to output both precursors and operations simultaneously, such predictions collapse material selection and process design into a single, opaque step that is difficult to verify, adapt, or safely execute in practice [35]. By explicitly separating these decisions, the two-stage formulation better aligns with real-world material synthesis workflows, enabling more verifiable, and practically deployable planning.

Precursor Prediction (PP). This task aims to identify a set of starting materials required to synthesize a given target material. In this task, only the chemical composition of the target material m is provided, without any explicit structural information, and the goal is to predict an appropriate precursor set P. For example, when the target material is $\mathrm{La_{1.2}Ca_{1.8}Mn_2O_7}$, the task is to predict a precursor set $\{\mathrm{CaCO_3}, \mathrm{La_2O_3}, \mathrm{MnO_2}\}$. Details of the preprocessing procedure for precursors are provided in Appendix C.

Synthesis Operation Prediction (SOP). This task focuses on determining the sequence of processing steps required to synthesize the target material, given both the target material and its corresponding precursor set. We define the synthesis operation as the sequence $O = (o_1, o_2, \ldots, o_n)$, where each o_t is selected from a predefined set of seven canonical solid-state synthesis operations: *mixing*, *drying*, *heating*, *sintering*, *annealing*, *quenching*, and *shaping*. A valid synthesis route may involve repeating certain operations multiple times, and the objective is to predict the ordered sequence O. For instance, in the case of $\mathrm{La_{1.2}Ca_{1.8}Mn_2O_7}$ with the precursor set $\{\mathrm{CaCO_3}, \mathrm{La_2O_3}, \mathrm{MnO_2}\}$, a plausible synthesis sequence is *mixing* $\rightarrow$ *heating* $\rightarrow$ *mixing* $\rightarrow$ *shaping* $\rightarrow$ *sintering*. Details of the preprocessing of synthesis operations are provided in Appendix C.

3 Proposed Method

We develop task-specific LLM fine-tuning strategies for the two constituent subproblems of MSP: PP and SOP. To structure these tasks, we introduce a discrete material class variable that induces a decision-chain formulation (Section 3.1). We then describe the PP stage (Section 3.2) and SOP stage (Section 3.3). We further propose explicit conditioning with hierarchical precursor types, which reinforces precursor-related constraints in the decoder state (Section 3.3.1), and analyze why this design is effective (Section 3.5). Finally, we describe the MSP inference pipeline (Section 3.4), where the independently trained PP and SOP models are linked sequentially: precursors predicted by PP serve as conditioning inputs to SOP.

3.1 Decision Chain via Discrete Material Class Variable

Materials often exhibit synthesis-related characteristics that are correlated with their chemical classes [30, 37]. While a material group does not uniquely determine a specific synthesis pathway, materials belonging to different groups (e.g., oxides, phosphates, nitrides, sulfides) often exhibit systematic differences in precursor selection, decomposition behavior, and synthesis operations [32]. In practice, domain experts attempt to identify the chemical group of a target material to determine an appropriate synthesis strategy. Inspired by this expert reasoning process, we introduce a decision chain framework that utilizes the material group, denoted as G, as a discrete intermediate variable for both Precursor Prediction (PP) and Synthesis Operation Prediction (SOP). In the PP task, given a target material m, the model first predicts its material group and subsequently generates its precursor set P conditioned on the predicted material group. Similarly, for SOP, the model predicts the material group G based on the target material m and the predicted precursor set P before generating the synthesis operation sequence O. Accordingly, both tasks are formulated as structured decision chains that first infer the material group and then perform task-specific generation. To implement this decision chain, we curate input prompts that explicitly instruct the model to **"first classify its material group"** before proceeding to the PP/SOP. Consequently, while fine-tuning the LLM, it learns to perform material group classification as a prerequisite step. This approach introduces a task-specific inductive bias, effectively constraining the search space for precursors and operations to a chemically plausible subspace. Note that PP and SOP are implemented using independently fine-tuned LLMs, and are subsequently integrated within a coherent framework where the material group decision chain is applied to both stages.

3.2 Precursor Prediction (PP) Stage

For the PP stage, we design a task-specific prompt to fine-tune the LLM such that the model first classifies the material group G from the target material m and subsequently selects an appropriate precursor set P. Let X_{PP} denote the complete input prompt, which includes the task description and the chemical formula of the target material m. We train the model using a standard language modeling (LM) objective under teacher forcing over the entire output sequence, with the target sequence $\mathbf{y}_{\text{PP}} = (G, P)$. Formally, the training loss is given by $\mathcal{L}_{\text{LM}}^{\text{PP}} = -\sum_{t=1}^{|\mathbf{y}_{\text{PP}}|} \log p_\theta(y_t \mid X_{\text{PP}}, \mathbf{y}_{\text{PP}<t})$, where $\mathbf{y}_{\text{PP}<t}$ denotes the ground-truth prefix.

3.3 Synthesis Operation Prediction (SOP) Stage

3.3.1 Challenges of Precursor Utilization

In SOP, precursors play a pivotal role in determining synthesis operation behavior, as they provide particularly informative cues for materials that require complex, multi-step synthesis procedures [30, 37, 32]. Intuitively, incorporating the precursor set into the input context is therefore expected to improve prediction accuracy compared to conditioning solely on the target material. To corroborate this, we compare two simple conditioning strategies applied at the LLM input stage: **target-only** and **implicit conditioning**, across two backbone LLMs, Qwen-2.5-7B and LLaMA-3.1-8B. The **target-only** approach conditions the LLM on the task description and the target material only, while **implicit conditioning** additionally incorporates the precursor set alongside the target material information to guide the model. We evaluate SOP performance using Normalized Edit Distance (NED), which measures the token-level similarity between the predicted and ground-truth synthesis operation sequences (note that each synthesis operation is treated as a single token).

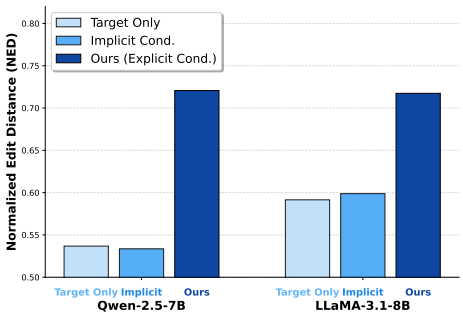


Figure 2: SOP performance under target-only, implicit, and explicit conditioning across two fine-tuned backbone LLMs.

However, as illustrated in Figure 2, contrary to our initial expectations, **implicit conditioning** does not consistently outperform the **target-only** baseline, despite the inclusion of additional precursor information. This observation suggests that merely providing precursor information as a part of the

LLM input is insufficient to ensure its effective utilization, especially in complex contexts—such as those involving long sequences or heterogeneous sequences of chemical formulas, where multiple precursor formulas are listed consecutively.

Therefore, this motivates the need to explicitly enforce the preservation and utilization of precursor-related information throughout the decoding process.

To effectively leverage the precursor information, we introduce an **explicit conditioning** strategy comprising two key components: (i) extracting hierarchical precursor types from the fine-grained precursor set to capture high-level behavioral constraints (Section 3.3.2), and (ii) enforcing conditioning via precursor constraint factorization (Section 3.3.3). As shown in Figure 2, this design leads to substantial and consistent performance gains across both backbone models, highlighting the importance of **explicit conditioning** over naive implicit input-level conditioning. Note that, since the PP and SOP modules are trained independently, we use the ground-truth precursor set to condition the SOP module during the model training.

3.3.2 Hierarchical Precursor Types

In SOP, it is important not only to identify fine-grained operational dependencies, such as the ordering of steps and the need for repeated operations, but also to capture higher-level synthesis trends that govern the overall process. For example, carbonate precursors typically require an initial heating step to remove CO_2 prior to phase formation [9]. This implies that the presence of a carbonate precursor provides a strong behavioral signal indicating the necessity of a heating step in the synthesis route. Building on this insight, we leverage a hierarchical abstraction of the precursor set in the form of precursor types, which capture key synthesis-relevant behaviors such as decomposition, volatility, and reactivity [32, 30, 37]. Let $P = \{p_1, \ldots, p_l\}$ denote the precursor set for a target material, and p_i denote the i-th precursor, where l is the number of precursors in P. Each precursor belongs to one of a finite set of precursor types (e.g., carbonate, nitrate, ammonium, phosphate, oxide, or other). We define a simple rule-based mapping function τ such that $Types = \tau(P)$, where $Types$ denotes the set of unique precursor types associated with the precursor set P. The resulting $Types$ provides a hierarchical, behaviorally meaningful abstraction of P that aligns closely with typical synthesis operation patterns. We denote the comprehensive precursor information as $Z := (Types, P)$. For instance, given a precursor set $P = \{\mathrm{CaCO_3}, \mathrm{La_2O_3}, \mathrm{MnO_2}\}$, the corresponding unique $Types$ would be $\{\text{carbonate}, \text{oxide}\}$.

3.3.3 Precursor Constraint Factorization

To ensure the effective utilization of precursor information beyond implicit input-level context, we introduce *precursor constraint factorization (PCF)*, which enforces the model to utilize the precursor information. Unlike naive implicit conditioning, which offers limited training pressure for precursor-specific representations in the decoder state, PCF introduces an explicit constraint that facilitates more consistent utilization of precursor information during synthesis operation generation. Concretely, SOP is formulated as learning a joint conditional distribution over (G, Z, O) given X_{SOP}, where $X_{\text{SOP}} = (\text{task prompt}, m, Z_{\text{in}})$ and Z_{in} denotes the ground-truth precursor information provided as part of the input. This induces the following factorization:

$$\begin{aligned} p_\theta(G, Z, O \mid X_{\text{SOP}}) = p_\theta(G \mid X_{\text{SOP}})\, p_\theta(Z \mid X_{\text{SOP}}, G) \\ \cdot\, p_\theta(O \mid X_{\text{SOP}}, G, Z). \end{aligned} \tag{1}$$

This formulation provides a principled decomposition that first enables the model, through the term $p_\theta(G \mid X_{\text{SOP}})$, to identify the material group as an intermediate variable corresponding to the decision-chain formulation (Section 3.1). The subsequent term $p_\theta(Z \mid X_{\text{SOP}}, G)$ explicitly constrains the model to predict and utilize precursor information. Finally, the term $p_\theta(O \mid X_{\text{SOP}}, G, Z)$ trains the model to generate the synthesis operation sequence conditioned on both the inferred material group and the predicted precursor variables. We implement this factorization by structuring the output sequence as $\mathbf{y}_{\text{SOP}} = (G, Z, O)$ and optimizing a standard autoregressive language modeling objective,

$$\mathcal{L}_{\text{LM}}^{\text{SOP}} = -\sum_{t=1}^{|\mathbf{y}_{\text{SOP}}|} \log p_\theta(y_t \mid X_{\text{SOP}}, \mathbf{y}_{\text{SOP}<t}). \tag{2}$$

During training, the ground truth precursor information Z_{in} is provided as part of the input context while simultaneously being modeled as an explicit random variable Z in the output sequence. This

requires the decoder state to remain sufficient for predicting Z even when Z is already observed in the input. Under this formulation, minimizing the SOP loss jointly constrains the model to accurately predict both Z and O, thereby discouraging solutions in which precursor-specific information is not retained in the decoder representation during operation generation. Consequently, during operation generation, the decoder state must remain aligned with precursor-specific constraints, enforcing retention and utilization of Z rather than treating precursor information as a passive component of the input context.

3.4 MSP Inference Pipeline

At inference time, MSP-LLM produces a synthesis plan from the target material composition m. Given this target, (1) the fine-tuned PP model first generates a ranked set of candidate precursor combinations. (2) Conditioned on a selected precursor set, the fine-tuned SOP model then produces coherent sequences of synthesis operations that specify how to realize the target material using the predicted precursors. By integrating these steps, the framework effectively achieves the MSP task.

3.5 Why Precursor Constraint Factorization Works

Setup. For clarity, we omit G and focus on the precursor information Z and the operation sequence O. We decompose the input context as $X_{\text{SOP}} = (X, Z_{\text{in}})$, where $X = (\text{task prompt}, m)$ and Z_{in} is the ground-truth precursor information provided in the prompt. We distinguish Z_{in} from the structured precursor variable $Z := (\text{Types}, P)$, which PCF treats as an explicit random variable in the output sequence. Implicit conditioning optimizes $-\log p_\theta(O \mid X_{\text{SOP}})$, whereas PCF optimizes $-\log p_\theta(Z, O \mid X_{\text{SOP}})$.

Marginalization under implicit conditioning. Since implicit conditioning does not require the model to distinguish between different precursor-specific generation mechanisms (i.e., Z_{in} and Z), it admits solutions that achieve low training loss while approximating the marginalized distribution $p(O \mid X) = \sum_z p(O \mid X, Z = z)\, p(Z = z \mid X)$, in which distinct precursor-conditioned patterns collapse into a single aggregated distribution and the decoder state is not required to retain instance-specific precursor information.

Precursor retention under PCF. The distinction between the marginalized distribution $p(O \mid X)$ and the precursor-conditioned distribution $p(O \mid X, Z)$ can be formalized using a basic property of conditional entropy: $H(O \mid X, Z) \leq H(O \mid X)$, with equality if and only if $O \perp Z \mid X$. When precursor information is informative for operations, i.e., $I(O; Z \mid X) > 0$, modeling $p(O \mid X, Z)$ admits strictly lower uncertainty than the marginalized predictor. Realizing this benefit, however, requires the decoder state to retain precursor information during operation generation. PCF provides this by training the model to predict Z as part of the output sequence, acting as an auxiliary objective that forces the decoder to retain precursor-specific features. This mitigates collapse into the marginalized predictor and narrows the gap between the learned model and $p(O \mid X, Z)$, yielding operation sequences that better reflect the given precursors.

Empirical verification. We further verify this mechanism by probing the decoder hidden states directly. We compare three settings: target-only (no precursors), implicit conditioning, and explicit conditioning with PCF; the latter two exclude hierarchical precursor types to isolate the effect of PCF. We extract mean-pooled hidden states during O generation and train a linear classifier for multi-label precursor type prediction (6 classes). As shown in Table 1, implicit conditioning improves over the target-only baseline by only 3.4 percentage points despite receiving precursor information as input, indicating that input-side precursor information is largely marginalized in the decoder state. In contrast, PCF yields a substantial 16.4 percentage-point gain over implicit conditioning, confirming that explicit prediction of precursor variables actively preserves precursor-specific information in the decoder state. Together with the consistent SOP gains in Figure 2, this provides direct evidence that PCF achieves the precursor-aware decoding behavior predicted

Table 1: Model probing accuracy for precursor types (Unseen formula, LLaMA-3.1-8B). Δ denotes improvement over the previous row.

Method	**Acc.**	Δ
Target only	0.6502	—
Implicit	0.6838	*+3.4 pp*
Explicit (PCF)	**0.8479**	**+16.4 pp**

by our analysis. A counterfactual precursor-substitution analysis (Appendix E.10) further shows that PCF makes the generated operations respond to precursor changes in a chemically consistent direction, whereas implicit conditioning largely does not.

4 Experiments

We evaluate the effectiveness of MSP-LLM across three settings: the PP task (Section 4.1), which predicts precursor sets from a target material; the SOP task (Section 4.2), which predicts synthesis operations given the target material and ground-truth precursors; and the MSP task (Section 4.3), which predicts both precursors and synthesis operations simultaneously from the target material composition.

Datasets. We utilize an inorganic material synthesis dataset that extracts synthesis recipes from materials science papers [19], which has been used in prior studies focusing on PP [24, 17]. To address the MSP task, we curate a subset of the data that contains full target, precursor, and synthesis operation information, resulting in 10,851 real-world synthesis records. For PP, we restrict the precursors to those with a frequency of at least five, yielding a set of 288 precursors. For synthesis operations, the original scheme consists of five categories, namely *mixing*, *quenching*, *heating*, *drying*, and *shaping*. In the SOP setting, we refine this scheme by decomposing *heating*, which conflates procedurally distinct operations, into *sintering*, *annealing*, and a general *heating* category, yielding seven operation types that better reflect real-world synthesis procedures. Additional details about the dataset are provided in Appendix C.

Implementation Details. We construct MSP-LLM by fine-tuning LLMs to perform both PP and SOP. We leverage open-source models LLaMA-3.1-8B [10] and Qwen 2.5-7B [29], and apply QLoRA [5] to enable efficient fine-tuning with limited computational resources. As detailed in Section 3.4, we fine-tune and evaluate the LLMs independently for each task. For the MSP task, these separately fine-tuned models are deployed simultaneously to generate the final output. Additional implementation details are provided in Appendix B.

Evaluation Protocol. We consider two dataset split strategies with an 8:1:1 train/validation/test ratio for experiments. Each sample is associated with a unique precursor set and synthesis operation sequence. The first is a random split, in which target formulas may appear across different splits, whereas each occurrence corresponds to a different precursor set and synthesis operation sequence due to the one-to-many mapping. The second is a target-disjoint split, where target formulas are strictly separated across different splits, enabling evaluation on previously unseen target materials. Additional details are provided in Appendix C. For SOP, all methods additionally receive synthesis context extracted from the titles of source papers (Appendix C.1); results without this context are reported in Appendix E.1.

4.1 Precursor Prediction (PP) Task

Baseline Methods. We prepare two types of baselines: (1) Specialized models for PP: we consider retrieval-based approaches that leverage reference material information through retrieval mechanisms, including the method of He et al. [13], which uses a composition-vector–based retrieval scheme, and Retrieval-Retro [24], a state-of-the-art model that combines advanced retrieval mechanisms with attention. (2) LLM baselines: we follow the prompt design of Prein et al. [28] to evaluate proprietary LLMs on PP, and conduct experiments with GPT-4o-mini, GPT-4o, GPT-5.1, and Claude Haiku 4.5. Here, we report the performance of MSP-LLM using an LLM that is independently trained specifically for the PP task.

Evaluation Protocol. Following prior work [28, 24], we adopt the Top-K Exact Match metric to evaluate precursor prediction performance.

Empirical Results. In Table 2, we observe that both specialized models and API-based LLMs exhibit limited capability in precursor prediction. Specifically, while the retrieval-based specialized model (Retrieval-Retro) achieves performance comparable to GPT-4o, it relies on an external retrieval system and lacks flexibility in adapting the size of the predicted precursor set, which constrains its overall accuracy. Moreover, while increasing model scale and the number of in-context examples improves API-based LLMs, a substantial performance gap remains. In contrast, our framework, regardless of the underlying LLM backbone, consistently outperforms all baselines across all Top-K levels. Since these LLMs are evaluated with few-shot prompting, we further compare against GPT-4o-mini

Table 2: PP performance on two dataset splits.

Method	Split 1				Split 2			
	Top-1	Top-3	Top-5	Top-10	Top-1	Top-3	Top-5	Top-10
Specialized Models								
He et al. [13]	53.11	60.05	61.06	63.07	54.01	61.10	62.69	65.02
Retrieval-Retro [24]	60.88	69.38	70.93	74.13	66.32	74.16	75.84	78.54
Large Language Models (API, few-shot prompting)								
GPT-4o-mini (20-shot)	56.58	65.36	67.82	70.66	54.85	63.40	67.91	70.34
GPT-4o-mini (40-shot)	55.21	63.44	66.27	69.10	55.97	64.27	67.82	71.36
GPT-4o (20-shot)	60.05	68.56	71.76	75.05	60.17	69.31	73.04	76.31
GPT-4o (40-shot)	61.79	70.48	73.40	76.32	61.57	71.27	73.97	77.05
GPT-5.1 (20-shot)	64.72	77.42	80.53	83.27	67.07	77.61	82.37	85.17
GPT-5.1 (40-shot)	64.08	76.33	79.62	83.46	66.60	77.89	82.74	86.01
Claude Haiku 4.5 (20-shot)	55.03	68.37	74.50	78.88	56.44	68.84	73.13	78.92
Claude Haiku 4.5 (40-shot)	57.31	68.74	74.31	78.43	56.53	71.64	76.12	81.81
MSP-LLM (Qwen-2.5-7B)	**71.02**	85.37	90.49	94.06	71.92	88.62	92.72	95.53
MSP-LLM (LLaMA-3.1-8B)	70.75	**85.56**	**91.68**	**94.97**	**72.85**	**88.99**	**93.56**	**96.36**

Table 3: SOP performance on two dataset splits.

Method	Split 1				Split 2			
	Top-10	NED↑	LCS↑	F-1↑	Top-10	NED↑	LCS↑	F-1↑
Specialized Models								
Retrieval based (He et al. [13])	10.4	0.6072	0.7194	0.7596	11.19	0.6030	0.7198	0.7564
SPENDE [23]	N/A	0.4508	0.5824	0.6013	N/A	0.4522	0.5821	0.6010
Large Language Models (API, few-shot prompting)								
GPT-4o-mini (20-shot)	5.85	0.5717	0.6992	0.7355	7.09	0.5847	0.7089	0.7407
GPT-4o-mini (40-shot)	5.94	0.5844	0.7092	0.7454	7.93	0.5934	0.7141	0.7442
GPT-4o (20-shot)	5.03	0.5654	0.6925	0.7219	3.82	0.5670	0.6865	0.7166
GPT-4o (40-shot)	4.34	0.5693	0.6890	0.7195	4.94	0.5749	0.6924	0.7245
GPT-5.1 (20-shot)	2.93	0.5334	0.6583	0.7101	2.33	0.5301	0.6547	0.6968
GPT-5.1 (40-shot)	2.74	0.5434	0.6660	0.7146	2.33	0.5261	0.6498	0.6925
Claude Haiku 4.5 (20-shot)	2.56	0.5534	0.6727	0.7174	1.68	0.5439	0.6656	0.7017
Claude Haiku 4.5 (40-shot)	4.11	0.5669	0.6801	0.7172	3.26	0.5569	0.6730	0.7037
MSP-LLM (Qwen-2.5-7B)	**26.51**	**0.7207**	**0.8100**	**0.8359**	**32.18**	0.7420	**0.8240**	0.8493
MSP-LLM (LLaMA-3.1-8B)	**26.51**	0.7174	0.8075	0.8353	31.44	**0.7424**	0.8235	**0.8498**

fine-tuned on the same training data, over which MSP-LLM still achieves higher performance on PP, SOP, and MSP (Appendix E.3). These results demonstrate that structured, task-specific fine-tuning and domain-aware inductive biases enable more effective learning of precursor selection constraints than either prompt-based conditioning or retrieval-dependent pipelines.

4.2 Synthesis Operation Prediction (SOP) Task

Baseline Methods. We evaluate two categories of baselines: (1) Specialized models for SOP: to emulate real experimental practices of chemists, who search for materials similar to a target and adapt their reported synthesis procedures, we use the retriever proposed by He et al. [13] to identify similar materials in the training set and take their associated synthesis operations as predictions. In addition, we evaluate SPENDE [23] as a specialized baseline. (2) LLM baselines: we design task-specific prompts to evaluate proprietary LLMs on SOP, and conduct experiments with GPT-4o-mini, GPT-4o, GPT-5.1, and Claude Haiku 4.5.

Evaluation Protocol. Since SOP is a sequence generation task, we adopt the Top-K Exact Match metric as a strict evaluation criterion that requires both the operation set and the exact ordering of the sequence to match the ground truth. To provide a more fine-grained assessment, we additionally report the Normalized Edit Distance (NED), which measures the similarity between predicted and ground-truth operation sequences, and the Longest Common Subsequence (LCS), which captures the length of the longest ordered subsequence shared with the ground truth. Finally, to evaluate content-level coverage independent of ordering, we use the multiset F-1 score (F-1), which measures the overlap between the predicted and reference operation elements while accounting for duplicates. For these sequence-level metrics, we report the maximum value among the Top-10 candidate predictions. For SPENDE [23], which produces only a single output per input, top-10 metrics are not applicable; instead, we report alternative evaluation metrics computed on the single generated output. We report only the Top-10 results in the main table, while the Top-1, Top-3, and Top-5 metrics are provided in Appendix H.

Empirical Results. Table 3 summarizes the results. Overall, LLM-based baselines exhibit limited performance in the SOP setting. We observe no consistent improvement with increasing model size; instead, smaller models such as GPT-4o-mini achieve relatively better performance compared to larger LLMs. This suggests that extensive general knowledge about target materials may act as noisy information when generating structured synthesis operation sequences, in the absence of an appropriate conditioning or decision strategy, rather than providing a clear benefit. SPENDE [23], which is specifically designed for thermoelectric materials, shows low performance on more general materials, indicating that it is not well-suited for SOP prediction in broad chemical spaces. The strongest baseline performance is obtained when retrieval-based methods [13] are employed, which more closely resemble the workflow of experimental chemists. Although this approach outperforms other models, it remains heavily dependent on the retrieved synthesis procedures of similar materials and therefore exhibits limited predictive generalization. By contrast, the proposed method consistently and substantially outperforms all baselines across all splits and evaluation metrics. This improvement can be attributed to the material group decision chain, which enables the model to select synthesis strategies tailored to specific material classes, as well as to the use of explicit conditioning that allows precursor information to be effectively leveraged during synthesis operation prediction.

Table 4: MSP Performance on two dataset splits.

Method	Split 1				Split 2			
	Top-1	Top-3	Top-5	Top-10	Top-1	Top-3	Top-5	Top-10
Large Language Models (API, few-shot prompting)								
GPT-4o-mini (20-shot)	0.0	0.09	0.09	0.19	0.19	0.28	0.28	0.47
GPT-4o-mini (40-shot)	0.0	0.0	0.09	0.09	0.27	0.27	0.37	0.64
GPT-4o (20-shot)	0.27	0.55	0.64	0.73	0.47	0.47	0.47	0.56
GPT-4o (40-shot)	0.27	0.27	0.37	0.64	0.37	0.84	0.93	1.12
GPT-5.1 (20-shot)	0.46	0.64	0.73	1.01	0.65	0.75	0.75	0.75
GPT-5.1 (40-shot)	0.55	0.64	0.64	0.73	0.37	0.37	0.47	0.47
Claude Haiku 4.5 (20-shot)	0.56	0.65	0.65	0.74	0.56	0.65	0.65	0.75
Claude Haiku 4.5 (40-shot)	0.56	0.65	0.65	0.74	0.37	0.84	0.93	1.12
GPT-5.1(PP) + GPT-4o-mini (SOP)	0.91	1.65	2.01	3.75	1.4	2.71	3.64	5.41
MSP-LLM (Qwen-2.5-7B)	**5.76**	**10.79**	**13.62**	**18.56**	6.62	**13.34**	**16.79**	22.48
MSP-LLM (LLaMA-3.1-8B)	4.75	9.69	13.35	18.19	**6.72**	12.12	16.60	**23.23**

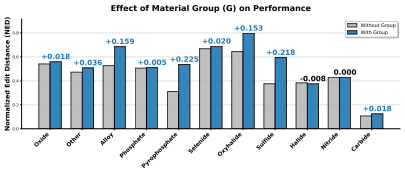


Figure 3: Effect of the material group decision chain on performance (NED) across 11 material classes.

4.3 Material Synthesis Planning (MSP) Task

Baseline Methods. Since no existing methods directly address the MSP task, we construct several baseline pipelines. For LLM-based baselines, we prompt a single LLM to perform the entire MSP task in a single process. Additionally, we combine the best-performing LLMs for each subtask into a hybrid pipeline for comparison. Note that MSP-LLM employs LLMs that are independently fine-tuned for the PP and SOP task.

Evaluation Protocol. To evaluate MSP, both the predicted precursor set and synthesis operation sequence must be correct. For a rigorous assessment, we adopt the Top-K Exact Match metric, which counts a prediction as correct only when both subtasks, PP and SOP, are simultaneously matched to the ground truth. For both our MSP-LLM and the LLM-based hybrid pipeline, SOP is conditioned on the top-1 precursor prediction during evaluation.

Empirical Results for MSP. Table 4 summarizes the results. All LLM-based baselines struggle on MSP. With the exception of the hybrid pipeline (GPT-5.1 + GPT-4o-mini; each performs the best on PP and SOP, respectively), none of the methods exceeds 2% in terms of Top-10 accuracy. This indicates that predicting both precursors and synthesis operations jointly with a single model is highly challenging, and that a two-stage pipeline can provide some benefit. Nevertheless, even the hybrid pipeline achieves only 3.75% and 5.41% on the two splits, respectively. Despite this difficulty, the proposed method attains substantially higher performance, achieving Top-10 scores of 18.56% on Split 1 and 23.23% on Split 2. Since this strict criterion may penalize chemically valid alternative routes, we additionally provide a relaxed evaluation in Appendix E.5, a chemical validity evaluation in Appendix E.4, an error propagation analysis of the MSP pipeline in Appendix E.2, extended comparisons with additional baselines in Appendix E.3 and per-class performance in Appendix E.6.

4.4 Further Analyses

Ablation Studies. To evaluate the contribution of each component in the MSP subtasks, we perform ablation studies in Tables 5 (a) and (b). For both PP and SOP, removing the material group decision chain leads to a noticeable performance degradation, highlighting the importance of structured decision-making for both tasks. For SOP, we further examine the effects of explicit conditioning and the precursor type information. We observe that explicit conditioning and the precursor types play a critical role in the observed performance gains, underscoring the importance of incorporating precursor constraints in synthesis operation prediction.

Moreover, to examine the effectiveness of the material group decision chain, we analyze how it improves performance across the 11 material classes. As shown in Figure 3, simply introducing the decision chain yields consistent performance gains in the majority of classes (9 out of 11). Notably, although oxides constitute approximately 85% of the dataset, the decision chain provides chemically plausible guidance across different material categories, rather than biasing predictions toward the dominant class. The group classification performance is reported in Appendix E.7.

Table 5: Ablation studies on PP and SOP (Split 1). EC: explicit conditioning with precursor types.

(a) Precursor Prediction (PP)

Model	G	Top-1	Top-3	Top-5	Top-10
(A) Naive Fine-tuning		70.59	84.90	90.62	94.39
(B) + Material Group (**MSP-LLM**)	✓	**70.75**	**85.56**	**91.68**	**94.97**

(b) Synthesis Operation Prediction (SOP)

Model	G	EC	Top-10	NED↑	LCS↑	F-1↑
(A) Naive Fine-tuning			11.24	0.5336	0.6311	0.6572
(B) + Material Group	✓		13.35	0.5623	0.6605	0.6864
(C) + Explicit Conditioning		✓	25.13	0.7168	0.8064	0.8342
(D) G + EC w/o precursor types	✓	✓	25.50	0.7188	0.8067	0.8353
(E) MSP-LLM (Qwen-2.5-7B)	✓	✓	**26.51**	**0.7207**	**0.8100**	**0.8359**

Integration with Downstream Condition Prediction Models. As MSP-LLM produces both precursors and synthesis operations, a publicly available condition prediction model [15] that takes this information as input can be naturally combined with our framework to additionally predict temperature and duration for heating operations. On the target-disjoint split (Split 2), the integrated pipeline achieves a Mean Absolute Error (MAE) of 134.9 °C and 6.7 h for heating operations. Table 6 presents cases in which MSP-LLM successfully predicts the precursor set and synthesis operation sequence, and the specific heating conditions are generated via the integrated framework. Overall, the resulting plan provides an informative starting point for synthesis planning of target materials. Additional cases are provided in Appendix E.8.

Table 6: Synthesis plan generation on unseen materials. Cells show **predicted** / reported values from MSP-LLM integrated with the condition model [15]. DOIs in Appendix.

#	Op.	T (P/R, °C)	t (P/R, h)	#	Op.	T (P/R, °C)	t (P/R, h)
Case 1: $Ba_{0.774}Ca_{0.226}Ti_{0.9}Sn_{0.1}O_3$				**Case 2:** $Nd_{0.7}Sr_{0.3}Mn_{0.9}Ga_{0.1}O_3$			
Pre.: $BaCO_3$, $CaCO_3$, SnO_2, TiO_2				*Pre.:* Ga_2O_3, MnO_2, Nd_2O_3, $SrCO_3$			
Ops: mix→dry→**heat**→mix→shape→**sinter**				*Ops:* mix→**heat**→mix→**heat**→shape→**sinter**			
0	heat	**1108** / 1200	**3.9** / 4.0	0	heat	**1066** / 1000	**24.3** / 24.0
1	sinter	**1347** / 1400	**3.4** / 4.0	1	heat	**1173** / 1350	**24.7** / 24.0
				2	sinter	**1398** / 1350	**22.4** / 24.0

5 Related Works

Material Synthesis Planning. MSP is a long-standing challenge in chemical sciences [33, 1]. Conventional approaches based on DFT [8, 14] require atomic structures of products and precursors, which are typically unavailable in real experiments [27, 4]. To bypass this requirement, recent machine learning methods for composition-based MSP have focused on either predicting precursor sets [13, 24] or generating sequences of synthesis operations [23]. However, no existing method addresses both subtasks within a single framework, leaving the MSP task unresolved.

LLMs for Materials Science. LLMs have been adopted in materials science to leverage their prior knowledge of physical chemistry, supporting downstream predictive and generative models [2, 22, 21, 38]. Building on advances in reasoning capability, recent works have begun to address MSP. RHS2LHS and TGT2CEQ [25] infer precursors and chemical reactions, but require ground-truth synthesis pathways as additional input and do not address SOP. LLM-QDMD [4] designs experiments for quantum dot synthesis, but is restricted to refining existing plans within a single material domain. A general LLM-based framework that performs both precursor and operation prediction across diverse materials remains unexplored. A more detailed discussion of related work is provided in Appendix A.

6 Conclusion

This paper proposes MSP-LLM, an LLM-based framework for MSP that decomposes the task into two subproblems: precursor prediction (PP) and synthesis operation prediction (SOP), each addressed through structured fine-tuning with a material group decision chain and explicit precursor conditioning. Our framework achieves 94.97% Top-10 accuracy on PP, an F-1 score of 0.8359 on SOP, and consistently outperforms baselines on the MSP task. We further show that integrating the framework with a condition prediction model yields synthesis plans including temperature and duration, providing an informative starting point for synthesis planning of target materials. These results highlight the practical potential of MSP-LLM for autonomous MSP, a long-standing challenge in chemical sciences.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2026-25607361). This research was also supported by the Fundamental Research Program (PNKB300) of the Korea Institute of Materials Science, funded by the Ministry of Science and ICT. Additional support was provided by a National Research Foundation of Korea (NRF) grant funded by the Ministry of Science and ICT (RS-2022-NR068758).

References

[1] Milad Abolhasani and Eugenia Kumacheva. The rise of self-driving labs in chemical and materials sciences. *Nature Synthesis*, 2(6):483–492, 2023.

[2] Adib Bazgir, Yuwen Zhang, et al. Matagent: A human-in-the-loop multi-agent llm framework for accelerating the material science discovery cycle. In *AI for Accelerated Materials Design-ICLR 2025*, 2025.

[3] Andres M Bran, Theo A Neukomm, Daniel P Armstrong, Zlatko Jončev, and Philippe Schwaller. Chemical reasoning in llms unlocks steerable synthesis planning and reaction mechanism elucidation. *arXiv preprint arXiv:2503.08537*, 2025.

[4] So Eun Choi, Miyoung Jang, SoHee Yoon, SangHyun Yoo, Jooyeon Ahn, Minho Kim, Ho-Gyeong Kim, Yebin Jung, Seongeon Park, Young-Seok Kim, et al. Llm-driven synthesis planning for quantum dot materials development. *Journal of Chemical Information and Modeling*, 65(6):2748–2758, 2025.

[5] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

[6] Qianggang Ding, Santiago Miret, and Bang Liu. Matexpert: Decomposing materials discovery by mimicking human experts. *arXiv preprint arXiv:2410.21317*, 2024.

[7] Yuanqi Du, Yingheng Wang, Yining Huang, Jianan Canal Li, Yanqiao Zhu, Tian Xie, Chenru Duan, John Gregoire, and Carla P Gomes. M^2 hub: Unlocking the potential of machine learning for materials discovery. *Advances in Neural Information Processing Systems*, 36:77359–77378, 2023.

[8] Eberhard Engel. *Density functional theory*. Springer, 2011.

[9] Amparo Fuertes. Synthetic approaches in oxynitride chemistry. *Progress in Solid State Chemistry*, 51:63–70, 2018.

[10] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[11] Nate Gruver, Anuroop Sriram, Andrea Madotto, Andrew Gordon Wilson, C Lawrence Zitnick, and Zachary Ulissi. Fine-tuned language models generate stable inorganic materials as text. *arXiv preprint arXiv:2402.04379*, 2024.

[12] Tanishq Gupta, Mohd Zaki, NM Anoop Krishnan, and Mausam. Matscibert: A materials domain language model for text mining and information extraction. *npj Computational Materials*, 8(1): 102, 2022.

[13] Tanjin He, Haoyan Huo, Christopher J Bartel, Zheren Wang, Kevin Cruse, and Gerbrand Ceder. Precursor recommendation for inorganic synthesis by machine learning materials similarity from scientific literature. *Science advances*, 9(23):eadg8180, 2023.

[14] Riley J Hickman, Malcolm Sim, Sergio Pablo-García, Gary Tom, Ivan Woolhouse, Han Hao, Zeqing Bao, Pauric Bannigan, Christine Allen, Matteo Aldeghi, et al. Atlas: a brain for self-driving laboratories. *Digital Discovery*, 4(4):1006–1029, 2025.

[15] Haoyan Huo, Christopher J Bartel, Tanjin He, Amalie Trewartha, Alexander Dunn, Bin Ouyang, Anubhav Jain, and Gerbrand Ceder. Machine-learning rationalization and prediction of solid-state synthesis conditions. *Chemistry of Materials*, 34(16):7323–7336, 2022.

[16] Martin Jansen. A concept for synthesis planning in solid-state chemistry. *Angewandte Chemie International Edition*, 41(20):3746–3766, 2002.

[17] Seongmin Kim, Yousung Jung, and Joshua Schrier. Large language models for inorganic synthesis predictions. *Journal of the American Chemical Society*, 146(29):19654–19659, 2024.

[18] Wan-Tae Kim, Weon-Gyu Lee, Hong-Eun An, Hiroyasu Furukawa, WooSeok Jeong, Sung-Chul Kim, Jeffrey R Long, Sohee Jeong, and Jung-Hoon Lee. Machine learning-assisted design of metal–organic frameworks for hydrogen storage: A high-throughput screening and experimental approach. *Chemical Engineering Journal*, 507:160766, 2025.

[19] Olga Kononova, Haoyan Huo, Tanjin He, Ziqin Rong, Tiago Botari, Wenhao Sun, Vahe Tshitoyan, and Gerbrand Ceder. Text-mined dataset of inorganic materials synthesis recipes. *Scientific data*, 6(1):203, 2019.

[20] Namkyeong Lee, Heewoong Noh, Sungwon Kim, Dongmin Hyun, Gyoung S Na, and Chanyoung Park. Density of states prediction of crystalline materials via prompt-guided multi-modal transformer. *Advances in Neural Information Processing Systems*, 36:61678–61698, 2023.

[21] Youjia Li, Vishu Gupta, Muhammed Nur Talha Kilic, Kamal Choudhary, Daniel Wines, Weikeng Liao, Alok Choudhary, and Ankit Agrawal. Hybrid-llm-gnn: integrating large language models and graph neural networks for enhanced materials property prediction. *Digital Discovery*, 4(2):376–383, 2025.

[22] Santiago Miret and NM Anoop Krishnan. Enabling large language models for real-world materials discovery. *Nature Machine Intelligence*, 7(7):991–998, 2025.

[23] Gyoung S Na. Artificial intelligence for learning material synthesis processes of thermoelectric materials. *Chemistry of Materials*, 35(19):8272–8280, 2023.

[24] Heewoong Noh, Namkyeong Lee, Gyoung S Na, and Chanyoung Park. Retrieval-retro: Retrieval-based inorganic retrosynthesis with expert knowledge. *Advances in Neural Information Processing Systems*, 37:25375–25400, 2024.

[25] Ryotaro Okabe, Zack West, Abhijatmedhi Chotrattanapituk, Mouyang Cheng, Denisse Córdova Carrizales, Weiwei Xie, Robert J Cava, and Mingda Li. Large language model-guided prediction toward quantum materials synthesis. *arXiv preprint arXiv:2410.20976*, 2024.

[26] Elton Pan, Soonhyoung Kwon, Sulin Liu, Mingrou Xie, Alexander J Hoffman, Yifei Duan, Thorben Prein, Killian Sheriff, Yuriy Roman-Leshkov, Manuel Moliner, et al. Diffsyn: A generative diffusion approach to materials synthesis planning. *arXiv preprint arXiv:2509.17094*, 2025.

[27] Thorben Prein, Elton Pan, Sami Haddouti, Marco Lorenz, Janik Jehkul, Tymoteusz Wilk, Cansu Moran, Menelaos Panagiotis Fotiadis, Artur P Toshev, Elsa Olivetti, et al. Retro-rankin: A ranking-based approach for inorganic materials synthesis planning. *arXiv preprint arXiv:2502.04289*, 2025.

[28] Thorben Prein, Elton Pan, Janik Jehkul, Steffen Weinmann, Elsa Olivetti, and Jennifer LM Rupp. Language models enable data-augmented synthesis planning for inorganic materials. *ACS Applied Materials & Interfaces*, 2025.

[29] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL `https://arxiv.org/abs/2412.15115`.

[30] Chintamani Nagesa Ramachandra Rao and Jagannatha Gopalakrishnan. *New directions in solid state chemistry*. Cambridge University Press, 1997.

[31] Alice Rueda, Mohammed S Hassan, Argyrios Perivolaris, Bazen G Teferra, Reza Samavi, Sirisha Rambhatla, Yuqi Wu, Yanbo Zhang, Bo Cao, Divya Sharma, et al. Understanding llm scientific reasoning through promptings and model's explanation on the answers. *arXiv preprint arXiv:2505.01482*, 2025.

[32] Ulrich S Schubert and Nicola Hüsing. *Synthesis of inorganic materials*. John Wiley & Sons, 2019.

[33] Nathan J Szymanski, Bernardus Rendy, Yuxing Fei, Rishi E Kumar, Tanjin He, David Milsted, Matthew J McDermott, Max Gallant, Ekin Dogus Cubuk, Amil Merchant, et al. An autonomous laboratory for the accelerated synthesis of novel materials. *Nature*, 624(7990):86–91, 2023.

[34] Sara Szymkuć, Ewa P Gajewska, Tomasz Klucznik, Karol Molga, Piotr Dittwald, Michał Startek, Michał Bajczyk, and Bartosz A Grzybowski. Computer-assisted synthetic planning: the end of the beginning. *Angewandte Chemie International Edition*, 55(20):5904–5937, 2016.

[35] Xiangru Tang, Qiao Jin, Kunlun Zhu, Tongxin Yuan, Yichi Zhang, Wangchunshu Zhou, Meng Qu, Yilun Zhao, Jian Tang, Zhuosheng Zhang, et al. Risks of ai scientists: prioritizing safeguarding over autonomy. *Nature Communications*, 16(1):8317, 2025.

[36] Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, et al. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60, 2023.

[37] Anthony R West. *Solid state chemistry and its applications*. John Wiley & Sons, 2022.

[38] Tianren Zhang and Dai-Bei Yang. Multimodal machine learning with large language embedding model for polymer property prediction. *Chemistry of Materials*, 37(18):7002–7013, 2025.

Supplementary Material

- MSP-LLM: A Structured Large Language Model Framework for Inorganic Material Synthesis Planning -

---

A Extended Related Works **15**

B Implementation Details **15**

B.1 Training Details 15

C Dataset **15**

C.1 Preprocessing 15

C.2 Dataset Split 16

D Baselines **16**

E Additional Experiments **17**

E.1 Synthesis Context Usage 17

E.2 Error Propagation Analysis in the MSP-LLM Pipeline 17

E.3 Further Baseline Comparisons 18

E.4 Chemical Validity Evaluation 19

E.5 Relaxed MSP Performance of MSP-LLM 19

E.6 Per-class Performance Across PP, SOP, and MSP. 20

E.7 Material Group Classification Performance 20

E.8 Additional Case Studies with the Integrated Pipeline 20

E.9 Performance across Multiple Seeds 21

E.10 Counterfactual Precursor Substitution 21

F Limitations **22**

G Broader Impacts **22**

H Supplementary Results **22**

I Prompt Templates **24**

A Extended Related Works

Material Synthesis Planning. MSP is a long-standing challenge in chemical sciences due to the inherent complexity and uncertainty associated with chemical reactions [33, 1]. In materials science, calculation- and simulation-based approaches on physical chemistry, often grounded in density functional theory (DFT) [8], have been used to screen precursor candidates and to analyze thermodynamically stable synthesis pathways toward target product materials [14]. However, these conventional methods have faced a fundamental limitation in real-world applications: the atomic structures of product and candidate precursor materials, which are required for reliable first-principles calculations, are typically unavailable in most chemical experiments [27, 4]. To overcome this challenge, prior machine learning methods for composition-based MSP have focused on either predicting precursor sets [13, 24] or generating sequences of synthesis operations [23]. However, MSP remains unresolved because there is currently no computational method capable of solving the MSP task, which requires predicting both plausible precursors and corresponding synthesis pathways.

LLM for Materials Science. LLMs have been increasingly adopted in materials science to exploit their broad prior knowledge of physical chemistry, thereby supporting the construction of downstream predictive and generative models [2, 22]. In particular, LLMs have notably accelerated materials discovery by facilitating efficient high-throughput screening based on surrogate models that predict the physical and chemical properties of inorganic materials [21] and polymers [38]. With the advances in the reasoning capabilities of LLMs, they have recently emerged as a method for addressing the MSP task, which constitutes a next remaining challenge for materials discovery. In this direction, RHS2LHS and TGT2CEQ were proposed to infer precursor materials and chemical reactions for designing an initial stage of the material synthesis plan [25]. However, both RHS2LHS and TGT2CEQ require additional information about ground-truth synthesis pathways when predicting precursors and do not address the SOP tasks. LLM-QDMD is an LLM-driven MSP method for designing experiment plans for the synthesis of quantum dot materials [4]. Although LLM-QDMD achieved substantial success within the domain of quantum dot materials, its applicability to real-world applications is inherently limited because it focuses on refining existing synthesis plans for quantum dot materials rather than generating new synthesis plans across general materials spaces.

B Implementation Details

Implementation details of MSP-LLM are provided in this section.

B.1 Training Details

All models are fine-tuned using Python 3.10.18, PyTorch 2.3.1, Accelerate 1.10.1, and Hugging Face Hub 0.34.4 on 2 NVIDIA RTX A6000 GPUs. To accommodate resource-limited settings, we employ QLoRA for parameter-efficient fine-tuning. The training hyperparameter configurations for each task and LLM are reported in Table 7

Table 7: Hyperparameter configuration for PP and SOP across different LLM backbones and dataset splits.

LLM	Task	Split 1			Split 2		
		LoRA Rank (r)	LoRA Alpha	lr / bs	LoRA Rank (r)	LoRA Alpha	lr / bs
LLaMA-3.1-8B [10]	PP	16	32	1e-4 / 4	16	32	1e-4 / 2
LLaMA-3.1-8B [10]	SOP	16	32	1e-4 / 2	16	32	1e-4 / 2
Qwen-2.5-7B [29]	PP	16	32	1e-4 / 4	16	32	1e-4 / 4
Qwen-2.5-7B [29]	SOP	16	32	1e-4 / 2	16	32	1e-4 / 4

C Dataset

C.1 Preprocessing

We preprocess the real-world experimental inorganic synthesis dataset [19].

Precursor Prediction (PP). We filtered out samples that do not contain synthesis operation–related information and removed data associated with compositional systems. We further excluded cases in which the corresponding precursor set contained only a single precursor. In addition, we applied a frequency-based filtering over the full dataset, retaining only precursors that appear at least five times. As a result, 288 unique precursors remained in the final dataset.

Synthesis Operation Prediction (SOP). In addition to the five basic synthesis operations provided in the original dataset—heating, mixing, shaping, quenching, and drying—we refine the heating category to better reflect real-world experimental practice. Specifically, we introduce two subcategories, sintering and annealing, and classify the remaining cases as generic heating. To distinguish these categories, we leverage synthesis-related subkeywords in the dataset: samples containing subkeywords related to "sinter" are labeled as sintering, while those containing subkeywords related to "anneal" are labeled as annealing.

To capture the synthesis intent and strategy of experimentalists, as well as coarse-grained material information (e.g., thin films), we use the titles of the source literature as contextual cues (synthesis context). We extract the following categories from the titles using GPT-4o: 'host material', 'dopant or substitution', 'material class', 'functional property', 'composition control', and 'processing or stimulus'. When relevant information is not present, the corresponding field is set to None. Since 'processing or stimulus' contains information directly related to synthesis operations, we exclude this field from the model inputs to ensure a precise evaluation. Instead, only the extracted information from 'host material', 'dopant or substitution', 'material class', 'functional property', and 'composition control', is used by all baselines as well as MSP-LLM.

C.2 Dataset Split

Fundamentally, a single material can be synthesized using multiple precursors and multiple operations. In other words, a single chemical formula can correspond to multiple synthesis routes, which implies a one-to-many relationship. Through data preprocessing, we removed duplicates and kept only unique combinations of chemical formula–precursor–operation.

In Split 1, the dataset was randomly divided into train/validation/test sets with an 8:1:1 ratio, without restricting chemical formulas. As a result, the same chemical formula may appear in the train, validation, and test sets. However, because the precursors and operations that must be predicted in the PP and SOP tasks differ from those seen during training, the model still needs to infer different precursors and operations even for a chemical formula it has already encountered.

In Split 2, the dataset was also divided with an 8:1:1 ratio, but chemical formulas were enforced to be non-overlapping across the train, validation, and test sets. This setup allows evaluation of the model's ability to perform PP and SOP on chemical formulas that were completely unseen during training, thereby measuring its generalization capability with respect to chemical formulas.

D Baselines

LLM Baselines. We compare LLM baselines including GPT-4o-mini[2] , GPT-4o[3] , GPT-5.1[4] , and Claude Haiku 4.5[5] . All models are accessed via their respective APIs. The GPT models are configured with a temperature of 0.8, while Claude Haiku is set to 0.3.

Specialized Models for Precursor Prediction (PP). We compared our approach with a retriever-based method He et al. [13] that learns the relationship between target materials and precursors through masked precursor completion, as well as a more recent work, Retrieval Retro[24], which employs an advanced attention mechanism and incorporates thermodynamic relationships between targets and precursors to identify more feasible precursor candidates. For both methods, we used their official implementations available on GitHub. In addition, when solving PP with the LLM baselines, we directly used the prompt from [28].

Specialized Models for Precursor Prediction (SOP). Using the retriever from [13] that was employed for PP, we leveraged the synthesis routes of materials similar to the target material as

[2] https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/

[3] https://openai.com/index/hello-gpt-4o/

[4] https://openai.com/index/gpt-5-1/

[5] https://www.anthropic.com/claude/haiku

the predicted outputs, serving as a baseline that reflects the practices of real synthesis experts. In addition, we adopted SPENDE[23] which is specifically designed for thermoelectric material synthesis prediction, using the official implementation provided on GitHub. The prompt used by the LLM baselines for the SOP task is provided in the Appendix I.

E Additional Experiments

E.1 Synthesis Context Usage

We examine the contribution of synthesis context on Split 2 by comparing three settings: context omitted during both training and testing, omitted only at test time, and included during both. As described in Appendix C, context comprises five attributes extracted from paper titles; the operation-related "processing or stimulus" field is excluded. PP uses the target composition, while SOP additionally receives precursors and, when available, context. Standalone SOP uses ground-truth precursors; MSP uses the Top-1 PP prediction. Thus, context-free MSP requires only the target composition as external input.

Table 8: Effect of synthesis context on Split 2. Yes/No indicates whether context is provided during training or testing. Top-K exact-match scores are percentages.

Context		**Qwen-2.5-7B**				**LLaMA-3.1-8B**			
Train	Test	**(a) Synthesis Operation Prediction (SOP)**							
		Top-10	NED↑	LCS↑	F-1↑	Top-10	NED↑	LCS↑	F-1↑
No	No	29.94	.732	.816	.843	29.38	.728	.816	.844
Yes	No	29.66	.730	.815	.839	28.92	.726	.812	.836
Yes	Yes	32.18	.742	.824	.849	31.44	.742	.824	.850
		(b) Material Synthesis Planning (MSP)							
		Top-1	Top-3	Top-5	Top-10	Top-1	Top-3	Top-5	Top-10
No	No	6.81	12.31	15.11	20.71	5.97	11.29	14.65	20.06
Yes	No	6.53	11.29	14.46	20.06	5.13	11.19	15.39	20.15
Yes	Yes	6.62	13.34	16.79	22.48	6.72	12.12	16.60	23.23

Table 8 shows that removing context from both training and testing reduces SOP Top-10 by 2.24 and 2.06 percentage points for Qwen and LLaMA, respectively. MSP Top-10 remains 20.71% and 20.06%, compared with 22.48% and 23.23% with context. Context therefore provides useful information, but is not required to generate synthesis plans. This limited sensitivity is consistent with the nature of the context: the extracted fields are already sparse, with 23.3%, 22.1%, and 22.8% missing in the training, validation, and test sets (up to 38.2% for "Dopant or substitution"), and they are only indirectly related to synthesis operations.

E.2 Error Propagation Analysis in the MSP-LLM Pipeline

Table 9: Error propagation analysis on Split 2 (LLaMA-3.1-8B). (*Left*) Comparison between MSP-LLM and an oracle SOP that always uses ground-truth precursors as input. (*Right*) Decomposition of MSP errors into four cases based on PP and SOP correctness.

(a) Comparison with Oracle SOP.

	Top-1	**Top-3**	**Top-5**	**Top-10**
MSP-LLM	6.72	12.12	16.60	23.23
Oracle SOP	9.20	16.62	22.73	31.80
Gap	2.48 pp	4.50 pp	6.13 pp	8.57 pp

(b) Error decomposition (Top-10).

Case	**Count**	**Ratio**
A: PP✓ SOP✓ (MSP success)	249	23.2%
B: PP✓ SOP✗ (SOP bottleneck)	534	49.8%
C: PP✗ SOP✓ (PP bottleneck)	89	8.3%
D: PP✗ SOP✗ (both failed)	200	18.7%

To understand where errors arise in the MSP-LLM pipeline, we conduct two complementary analyses. First, Table 9 compares MSP-LLM with an oracle SOP that always uses ground-truth precursors as

input; the resulting Top-10 gap of 8.57 pp quantifies the contribution of PP errors to overall MSP failure. Second, Table 9b decomposes MSP failures into four cases based on PP and SOP correctness. Case B (PP correct but SOP fails) accounts for the largest share at 49.8%, indicating that SOP is the primary bottleneck of the joint pipeline. Together, these results suggest that improving SOP offers the largest opportunity for advancing MSP, while reducing PP errors provides a complementary 8.57 pp room for further gain.

E.3 Further Baseline Comparisons

Table 10: PP, SOP, and MSP performance on Split 2.

	PP				SOP				MSP			
Method	**Top-1**	**Top-3**	**Top-5**	**Top-10**	**Top-10**	**NED↑**	**LCS↑**	**F-1↑**	**Top-1**	**Top-3**	**Top-5**	**Top-10**
Large Language Models (API, few-shot prompting)												
GPT-4o-mini (20-shot)	54.85	63.40	67.91	70.34	7.09	0.5847	0.7089	0.7407	0.19	0.28	0.28	0.47
GPT-4o-mini (40-shot)	55.97	64.27	67.82	71.36	7.93	0.5934	0.7141	0.7442	0.27	0.27	0.37	0.64
GPT-4o-mini (60-shot)	55.69	65.02	68.75	72.20	10.54	0.6099	0.7310	0.7616	1.40	1.96	2.05	2.24
GPT-4o-mini (80-shot)	59.51	67.44	71.92	75.37	9.24	0.6048	0.7238	0.7551	0.93	1.21	1.31	1.40
GPT-4o-mini (100-shot)	60.54	68.28	71.64	74.35	9.70	0.6082	0.7274	0.7572	0.93	1.59	1.59	1.68
GPT-4o (20-shot)	60.17	69.31	73.04	76.31	3.82	0.5670	0.6865	0.7166	0.47	0.47	0.47	0.56
GPT-4o (40-shot)	61.57	71.27	73.97	77.05	4.94	0.5749	0.6924	0.7245	0.37	0.84	0.93	1.12
GPT-4o (60-shot)	62.78	72.20	75.09	77.05	6.53	0.5888	0.7093	0.7398	0.75	0.93	0.93	1.03
GPT-4o (80-shot)	63.15	72.67	76.12	78.82	6.81	0.5883	0.7098	0.7392	0.47	0.93	1.03	1.12
GPT-4o (100-shot)	63.72	71.46	75.28	78.36	8.02	0.5926	0.7139	0.7430	0.56	0.84	1.03	1.31
Large Language Models (API, fine-tuned)												
GPT-4o-mini (fine-tuning)	69.78	79.48	81.90	86.10	15.11	0.6691	0.7727	0.8048	1.77	4.20	6.72	12.03
MSP-LLM (Qwen-2.5-7B)	71.92	88.62	92.72	95.53	**32.18**	0.7420	**0.8240**	0.8493	6.62	**13.34**	**16.79**	22.48
MSP-LLM (LLaMA-3.1-8B)	**72.85**	**88.99**	**93.56**	**96.36**	31.44	**0.7424**	0.8235	**0.8498**	**6.72**	12.12	16.60	**23.23**

We further conduct additional experiments with GPT-4o-mini and GPT-4o on PP, SOP, and MSP tasks. We extend the few-shot prompting setup to 60, 80, and 100 shots, and additionally fine-tune GPT-4o-mini on the same training data using naive fine-tuning (without MSP-LLM's task-specific strategies).

For few-shot prompting, performance slightly improves as the number of shots increases but quickly saturates and remains substantially below MSP-LLM, while also introducing practical challenges such as expanded context windows and increased computational cost. For fine-tuning, while GPT-4o-mini shows substantial improvement over its few-shot counterpart, our framework still achieves consistently higher performance across all tasks and metrics, indicating that the advantage of MSP-LLM persists even against a fine-tuned proprietary LLM.

We compare three additional MSP baselines on Split 2: naive two-stage fine-tuning and end-to-end fine-tuning using the same LLaMA-3.1-8B backbone, and nearest-neighbor retrieval. All settings use synthesis context. The retrieval baseline identifies similar training instances using target composition and synthesis context with MatSciBERT [12], and returns their precursor sets and operation sequences. These comparisons examine whether the gains arise merely from fine-tuning on the same data or retrieving similar synthesis routes.

Table 11: Further MSP baseline comparisons on Split 2 (exact match, %). Fine-tuned models use LLaMA-3.1-8B; all settings use synthesis context.

Method	Top-1	Top-3	Top-5	Top-10
Naive two-stage fine-tuning	4.1	8.3	9.9	12.7
End-to-end fine-tuning	4.8	10.8	13.0	16.1
Nearest-neighbor	5.8	9.8	10.3	10.8
MSP-LLM	6.72	12.12	16.60	23.23

Table 11 shows that MSP-LLM outperforms these baselines at every cutoff, reaching 23.23% Top-10 accuracy versus 12.7%, 16.1%, and 10.8%, respectively. End-to-end training improves over naive two-stage fine-tuning, while retrieval gains little as the candidate budget increases. Since naive two-stage fine-tuning shares the backbone, training data, and two-stage decomposition with MSP-LLM, the remaining gap reflects the contribution of the proposed structured fine-tuning strategies rather than fine-tuning on the same data alone.

E.4 Chemical Validity Evaluation

We complement exact-match evaluation with two chemical-validity checks on predicted precursor sets. *Element coverage* checks whether the precursors collectively contain the elements required by the target, while *mass balance* checks whether the predicted precursor set permits a mass-balanced reaction. Table 12 reports both criteria at Top-1, Top-3, Top-5, and Top-10 on Split 2.

Table 12: Chemical validity evaluation of PP on Split 2. Values are pass rates (%) for element coverage and mass balance.

	Element coverage				**Mass balance**			
Model	@1	@3	@5	@10	@1	@3	@5	@10
MSP-LLM (Qwen-2.5-7B)	99.6	99.9	100.0	100.0	99.4	99.8	99.9	100.0
MSP-LLM (LLaMA-3.1-8B)	99.6	100.0	100.0	100.0	99.3	99.8	99.8	100.0

Both models achieve pass rates above 99% at Top-1 and reach 100% at Top-10. These results complement exact-match evaluation by assessing basic chemical consistency of the predicted precursors. Such checks are necessary constraints, however, and do not by themselves establish experimental feasibility or validate the predicted operation sequence.

E.5 Relaxed MSP Performance of MSP-LLM

Table 13: Relaxed MSP performance of MSP-LLM (LLaMA-3.1-8B) on Split 2 with relaxed PP and SOP criteria. Percentages in parentheses denote the fraction of test samples whose predicted precursors pass the corresponding PP threshold. PP Set-F1 measures precursor-level partial credit; PP Elem-F1 measures element-level coverage.

PP Criterion	**SOP Condition**	**Top-1**	**Top-3**	**Top-5**	**Top-10**
PP Set-F1 Criterion					
$\geq$0.8 (75.0%)	SOP Exact	6.90	12.50	17.07	23.97
$\geq$0.8	SOP NED $\geq$0.8	12.31	20.24	25.00	32.74
$\geq$0.8	SOP LCS $\geq$0.8	17.82	30.88	38.06	49.35
$\geq$0.8	SOP F-1 $\geq$0.8	**21.36**	**36.01**	**43.47**	**55.60**
PP Elem-F1 Criterion					
$\geq$0.9 (91.2%)	SOP Exact	7.56	15.58	21.27	29.38
$\geq$0.9	SOP NED $\geq$0.8	14.55	25.19	31.25	40.58
$\geq$0.9	SOP LCS $\geq$0.8	21.46	37.87	47.29	60.63
$\geq$0.9	SOP F-1 $\geq$0.8	25.65	43.75	53.36	67.91
$\geq$0.8 (99.4%)	SOP Exact	8.12	16.51	22.57	31.44
$\geq$0.8	SOP NED $\geq$0.8	15.39	26.49	33.21	43.75
$\geq$0.8	SOP LCS $\geq$0.8	22.67	40.21	50.93	65.76
$\geq$0.8	SOP F-1 $\geq$0.8	**27.05**	**46.36**	**57.37**	**73.13**

The Top-K exact-match metric used for MSP requires both the precursor set and the synthesis operation sequence to match the ground truth exactly, which can penalize chemically valid alternative routes. To allow partial credit for predictions that are chemically reasonable but not exact, we introduce two relaxed PP criteria and three relaxed SOP criteria. For PP, we consider (i) *PP Set-F1*, which grants credit when the predicted precursor set has high overlap with the ground truth (F1 ≥ 0.8), and (ii) *PP Elem-F1*, which verifies whether the predicted precursors collectively supply all elements required by the target. For SOP, we apply soft thresholds: given an average ground-truth sequence length of 4.0 steps, NED≥ 0.8 corresponds to at most one step difference, LCS≥ 0.8 to at least $3/4$ operations in correct order, and F1≥ 0.8 to at least 80% operation overlap. These relaxed evaluations provide a chemically informed assessment of partial credit, highlighting MSP-LLM's practical utility in providing synthesis guidance.

E.6 Per-class Performance Across PP, SOP, and MSP.

Table 14: Per-class PP, SOP, and MSP performance (Split 1, LLaMA-3.1-8B).

Class	n	PP Top-10	SOP Top-10	MSP Top-10 (strict)	MSP Top-10 (NED≥0.8)	MSP Top-10 (LCS≥0.8)	MSP Top-10 (F1≥0.8)
oxide	938	95.3	25.59	18.0	29.00	45.52	51.28
alloy	39	100.0	38.46	35.9	43.59	53.85	58.97
phosphate	28	100.0	21.43	3.6	7.14	32.14	35.71
pyrophosphate	14	92.9	28.57	14.3	14.29	21.43	21.43
oxyhalide	9	66.7	44.44	11.1	11.11	22.22	22.22
selenide	10	100.0	10.00	10.0	10.00	80.00	90.00
sulfide	8	87.5	50.00	12.5	12.50	12.50	12.50
not_classified	42	88.1	38.10	23.8	23.81	30.95	47.62
carbide	1	100.0	0.00	0.0	0.00	100.0	100.0
nitride	1	0.0	0.00	0.0	0.00	0.00	0.00
halide	4	100.0	0.00	0.0	25.00	25.00	25.00

Table 15: Per-class PP, SOP and MSP performance (Split 2, LLaMA-3.1-8B).

Class	n	PP Top-10	SOP Top-10	MSP Top-10 (strict)	MSP Top-10 (NED≥0.8)	MSP Top-10 (LCS≥0.8)	MSP Top-10 (F1≥0.8)
oxide	907	96.9	30.10	21.9	31.75	48.18	54.47
alloy	61	100.0	45.90	42.6	42.62	52.46	57.38
phosphate	28	85.7	21.43	3.6	7.14	21.43	28.57
pyrophosphate	9	88.9	55.56	22.2	22.22	44.44	44.44
oxyhalide	11	81.8	18.18	27.3	36.36	45.45	45.45
selenide	7	100.0	42.86	42.9	42.86	57.14	71.43
sulfide	12	100.0	50.00	50.0	50.0	58.33	75.0
not_classified	32	87.5	37.50	21.9	21.88	40.62	46.88
carbide	4	100.0	50.00	50.0	50.0	75.0	75.0
nitride	1	100.0	0.00	0.0	0.0	100.0	100.0
halide	0	–	–	–	–	–	–

Tables 14 and 15 report per-class performance of MSP-LLM. Despite oxide dominance (∼85%), several minority classes achieve higher performance than oxide across PP, SOP, and MSP, indicating that the framework is not biased toward the majority class and generalizes effectively to diverse material groups.

E.7 Material Group Classification Performance

Table 16: Task-level accuracy and F-1 score on two dataset splits (LLaMA-3.1-8B).

Task	Split 1		Split 2	
	Accuracy	F-1	Accuracy	F-1
PP	96.0	0.83	96.9	0.80
SOP	95.4	0.72	96.4	0.78

We report the material group classification performance of the decision chain in Table 16. The classification accuracy of around 96% and F-1 scores ranging from 0.72 to 0.83 across both splits confirm that the decision chain learns meaningful chemical group information.

E.8 Additional Case Studies with the Integrated Pipeline

To further illustrate the practical applicability of the integrated pipeline, we provide three additional case studies in Table 17, complementing the two cases shown in Table 6. In each case, MSP-LLM successfully predicts the precursor set and operation sequence from the target composition, and the integrated condition prediction model [15] supplies the temperature and duration for each heating

Table 17: Additional case studies of synthesis plan generation on unseen materials. Cells show **predicted** / reported values from MSP-LLM integrated with the condition model [15].

#	Op.	T (P/R, °C)	t (P/R, h)
Case 3: $Mg_{0.94}Ni_{0.06}ZrNb_2O_8$ (DOI: 10.1007/s10854-017-7996-x)			
Pre.: MgO, Nb_2O_5, NiO, ZrO_2			
Ops: mix→dry→heat→mix→dry→mix→shape→sinter			
0	heat	**1026** / 1100	**4.5** / 2.0
1	sinter	**1351** / 1320	**4.9** / 4.0
Case 4: $Ni_{0.78}Cu_{0.20}Zn_{0.02}Fe_{1.96}O_{3.94}$ (DOI: 10.1016/j.jmmm.2017.12.108)			
Pre.: CuO, Fe_2O_3, NiO, ZnO			
Ops: mix→dry→heat→mix→dry→mix→shape→sinter			
0	heat	**856** / 800	**3.1** / 2.0
1	sinter	**989** / 900	**2.8** / 2.0
Case 5: $Pb_{0.95}Ba_{0.05}Li_{0.25}La_{0.25}Mo_{0.5}O_3$ (DOI: 10.1016/S0167-577X(00)00160-9)			
Pre.: $BaCO_3$, La_2O_3, Li_2CO_3, MoO_3, PbO			
Ops: mix→heat→mix→heat→mix→shape→sinter			
0	heat	**623** / 550	**11.6** / 12.0
1	heat	**677** / 650	**11.1** / 18.0
2	sinter	**786** / 650	**9.5** / 6.0

operation. For reference, the DOIs of the source publications for the two cases shown in the main paper (Table 6) are: Case 1, 10.1007/s10854-017-6960-0; Case 2, 10.1016/j.ssc.2007.09.002.

E.9 Performance across Multiple Seeds

Table 18 reports the mean and standard deviation over three seeds on Split 2 for PP, SOP, and MSP. Both backbones obtain similar average performance, with PP Top-10 above 95% and SOP Top-10 above 32%. MSP Top-10 reaches 23.48 ± 0.74 for Qwen and 23.91 ± 0.90 for LLaMA, showing consistent performance across the reported runs.

Table 18: Performance over three seeds on Split 2 (mean $\pm$ standard deviation). Top-K scores are percentages; NED, LCS, and F-1 are similarities.

(a) Precursor Prediction (PP)

Model	Top-1	Top-3	Top-5	Top-10
MSP-LLM (Qwen-2.5-7B)	72.73 ± 0.58	88.90 ± 0.20	93.38 ± 0.47	95.68 ± 0.22
MSP-LLM (LLaMA-3.1-8B)	73.23 ± 0.46	89.33 ± 0.36	93.59 ± 0.19	96.24 ± 0.12

(b) Synthesis Operation Prediction (SOP)

Model	Top-10	NED↑	LCS↑	F-1↑
MSP-LLM (Qwen-2.5-7B)	33.27 ± 0.77	0.7489 ± 0.0049	0.8283 ± 0.0031	0.8537 ± 0.0031
MSP-LLM (LLaMA-3.1-8B)	32.84 ± 1.14	0.7483 ± 0.0046	0.8286 ± 0.0039	0.8536 ± 0.0030

(c) Material Synthesis Planning (MSP)

Model	Top-1	Top-3	Top-5	Top-10
MSP-LLM (Qwen-2.5-7B)	6.96 ± 0.31	13.43 ± 0.35	17.26 ± 0.60	23.48 ± 0.74
MSP-LLM (LLaMA-3.1-8B)	7.00 ± 0.39	12.93 ± 0.58	17.04 ± 0.43	23.91 ± 0.90

E.10 Counterfactual Precursor Substitution

To examine whether PCF makes the decoder use the input precursors, we replace every carbonate precursor with the corresponding cation oxide (e.g., $SrCO_3$ →SrO) while keeping the target fixed (603 of 1,072 Split 2 test cases). Since carbonates typically require an initial heating step to remove CO_2 (Section 3.3.2), this substitution should reduce heating-family operations (heating, sintering, and

annealing). Following the probing analysis in Table 1, both models exclude hierarchical precursor types.

Table 19: Counterfactual carbonate→oxide substitution on Split 2 (LLaMA-3.1-8B, n=603). Heat ops: total number of heating-family operations over the 603 cases (original→substituted). $\downarrow / = / \uparrow$: heating count decreased/unchanged/increased among changed outputs.

Model	**Heat ops (total)**	**Δabs**	**Δrel**	**n changed**	**$\downarrow / = / \uparrow$ among changed (%)**
Implicit	945→909	-36	-3.8%	140	31 / 55 / 14
Explicit (PCF)	689→620	-69	-10.0%	128	52 / 39 / 9

As shown in Table 19, heating operations barely change under implicit conditioning (-3.8%) but decrease under PCF (-10.0%). Among the changed outputs, most implicit edits leave the heating count unchanged (55%), whereas most PCF edits reduce it (52% vs. 9% increase). Thus, PCF makes the generated operations respond to precursor changes in a chemically consistent direction, complementing the probing results in Table 1.

F Limitations

MSP-LLM successfully solves both PP and SOP and connects them to address the MSP task. However, in the current setup, only the Top-1 PP prediction is used as input, which means that errors in PP can propagate to the SOP stage. We provide a detailed error propagation analysis in Appendix E.2, which quantifies the contribution of PP errors to MSP failure. A strategy is needed to enable the use of multiple precursor candidates predicted by PP and to incorporate them into the SOP stage so that the model can select better precursor conditions and improve overall performance. In addition, since PP is restricted to precursors above a frequency threshold, its accuracy decreases for rare precursors, and generating precursors unseen during training remains difficult; a larger corpus with broader precursor coverage may alleviate this. Moreover, while MSP-LLM produces precursors and synthesis operations from a target material and can be integrated with downstream condition prediction models, it does not directly model processing conditions. Other synthesis attributes such as atmosphere and equipment are also outside the current scope. These aspects remain underexplored largely due to the lack of consistently curated data; once such data become available, incorporating them into the framework will be an important direction for future work.

G Broader Impacts

Positive Societal Impacts. MSP-LLM provides a structured LLM framework for autonomous material synthesis planning, predicting precursors and synthesis operations for a target material—tasks that are typically challenging and time-consuming for domain experts. By offering an informative starting point for synthesis, our framework can contribute to accelerating materials discovery and reducing the manual effort required in laboratory practice.

Negative Societal Impacts. The predicted synthesis plans should be reviewed by human experts before experimental execution, as model outputs may contain errors. We do not foresee additional risks beyond standard considerations for AI-assisted scientific tools.

H Supplementary Results

In this section, we provide additional metrics—Top-1, Top-3, and Top-5—supplementing the existing SOP task evaluation.

Table 20: Supplementary Results: Synthesis operation prediction (SOP) performance on two dataset splits.

Method	**Split 1**				**Split 2**			
	Top-1	**Top-3**	**Top-5**	**Top-10**	**Top-1**	**Top-3**	**Top-5**	**Top-10**
Specialized Models								
Retrieval based He et al. [13]	1.46	3.75	5.94	10.42	1.12	3.54	5.50	11.19
SPENDE [23]	3.66	N/A	N/A	N/A	5.71	N/A	N/A	N/A
Large Language Models (API)								
GPT-4o-mini (20-shot)	1.73	3.11	4.2	5.85	1.68	3.26	4.01	7.09
GPT-4o-mini (40-shot)	1.83	2.74	3.56	5.94	2.15	4.2	5.69	7.93
GPT-4o (20-shot)	0.09	1.09	2.56	5.03	0.47	1.03	1.77	3.82
GPT-4o (40-shot)	0.18	1.01	1.92	4.34	0.28	1.03	2.05	4.94
GPT-5.1 (20-shot)	0.73	1.55	1.65	2.93	0.75	1.31	1.4	2.33
GPT-5.1 (40-shot)	0.91	1.74	1.74	2.74	0.75	1.03	1.77	2.33
Claude Haiku 4.5 (20-shot)	0.18	0.73	1.56	2.56	0.47	0.84	1.21	1.68
Claude Haiku 4.5 (40-shot)	0.91	2.38	3.29	4.11	0.47	1.49	2.52	3.26
MSP-LLM (Qwen-2.5-7B)	**7.86**	**15.36**	**19.65**	**26.51**	**8.77**	**18.47**	22.57	**32.18**
MSP-LLM (LlaMA-3.1-8B)	6.40	13.71	19.20	**26.51**	8.12	17.35	**22.58**	31.44

I Prompt Templates

In this section, we present the prompt templates used to fine-tune LLMs for PP 3.2 and SOP 3.3. Additionally, we provide the prompts used for the LLM baseline for the SOP task.

Table 21: Prompt for Fine-tuning LLMs for Precursor Prediction (PP).

System Prompt: You are an expert solid-state chemist planning a material synthesis.

Prompt: Task:
Given the target compound {target_formula}, first classify its material group, and you must select the precursor needed to synthesize the target.

Typically, two or more precursors are required.

The possible material groups are:
`['oxide', 'alloy', 'phosphate', 'pyrophosphate', 'halide', 'oxyhalide', 'carbide', 'nitride', 'selenide', 'sulfide', 'other']`.

Output format:
```
[ ['material group'], ['precursor1', 'precursor2', ..., 'precursorN']
]
```

Return only the Python list of single-quoted strings with no extra text or explanation.

Table 22: Prompt for Fine-tuning LLMs for Synthesis Operation Prediction (SOP).

System Prompt: You are an expert solid-state chemist planning a material synthesis.

Prompt:
[PRECURSOR_TYPES]
types_present: {precursor types}
[/PRECURSOR_TYPES]

Task:
Given the target compound {target_formula}, first classify its material group, then based on the selected precursors {precursors}, propose one plausible synthesis operation sequence using only the following operations:
`['heating', 'sintering', 'annealing', 'mixing', 'shaping', 'quenching', 'drying']`.

Synthesis context: {keyword}

Typically, two or more precursors are required.

The possible material groups are:
`['oxide', 'alloy', 'phosphate', 'pyrophosphate', 'halide', 'oxyhalide', 'carbide', 'nitride', 'selenide', 'sulfide', 'other']`.

Output format:
`[ ['material group'], ['precursor_type1', ..., 'precursor_typeK'], ['precursor1', ..., 'precursorN'], ['op1', ..., 'opM'] ]`

Return only the Python list of single-quoted strings with no extra text or explanation.

Table 23: Prompt for LLM Baselines in Synthesis Operation Prediction (SOP).

System Prompt: You are an expert solid-state chemist specializing in synthesis operations.

Prompt: Given:
- Target material: {target_formula}
- Precursors for the target material: {precursors}

Task:
Propose a plausible synthesis operation sequence for preparing the target from the given precursors.

Instructions:
1) Choose operations only from the following allowed set: `{"heating", "sintering", "annealing", "mixing", "shaping", "quenching", "drying"}`.
2) "Heating" refers to general heating processes (e.g., calcination, firing) and explicitly excludes Sintering and Annealing.
3) Output only the operation names — do not include temperatures, times, atmospheres, parameters, or explanations.
4) Format the answer as a Python list of lists of strings **exactly**.
- Each inner list is a single synthesis operation sequence, e.g., `["mixing", "heating", "sintering"]`.
- Return **exactly {num_combinations} sequences**.
5) Each sequence must contain **one or more** operations; repetitions are allowed when chemically reasonable.
6) Do not invent new operation names, abbreviations, or comments. Do not output anything other than the list of lists.

Reference for operation semantics (do NOT include this section in the output):
- Heating: any general heating process (e.g., calcination, firing) except sintering and annealing.
- Sintering: high-temperature treatment to densify materials.
- Annealing: controlled heating/cooling for structural ordering or stress relief.

Synthesis context: {keyword}

Examples of synthesis operation sequences from target and its precursors:
{examples}

Output format (example shape only; return exactly {num_combinations} sequences):
```
[
["mixing", "heating", "sintering"],
["mixing", "drying", "heating", "annealing"],
...
]
```

Return only the data as a plain text Python list. Do not use Markdown code fences or language tags, and do not print escaped characters like \n. Start with [and end with]. Use single quotes for strings. No extra text.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state our contributions: a structured LLM-based framework (MSP-LLM) that addresses both PP and SOP within a single MSP framework, with a material group decision chain and precursor constraint factorization. The claims regarding superior performance over baselines on PP, SOP, and the joint MSP task are supported by the experimental results in Section 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of our framework in Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: N/A

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We include all the details for reproducing the experimental results in Section 4 and Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide a GitHub repository with the code and sufficient instructions for reproducing our results. The source code for MSP-LLM is available at https://github.com/HeewoongNoh/MSP-LLM.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We provide a description of the data splits in Section 4 and an anonymized GitHub repository with the code, sufficient instructions, and experimental details for reproducing our results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report the mean and standard deviation over three seeds for PP, SOP, and MSP on Split 2 with both LLaMA-3.1-8B and Qwen-2.5-7B in Appendix E.9 (Table 18).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide information on the necessary computational resources in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and confirm that our research conforms to it in every respect.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the potential positive societal impacts and negative impacts of our work in the Appendix G.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Although we use pretrained language models, they are only fine-tuned to output material synthesis planning predictions, so there are no significant risks of misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All external assets used in this work, including datasets, pretrained models, and baseline implementations, are properly cited and used in accordance with their respective licenses.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release the implementation of MSP-LLM as an anonymized GitHub repository, containing the code, training scripts, hyperparameter configurations, and instructions for reproducing our results.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve any crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This research does not involve human subjects or crowdsourcing.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were used only for grammar checking and editing during manuscript preparation, and did not contribute to the core methodology, scientific content, or originality of the research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.