A AtomWorld Setup Details

A.1 Supported action prompts

Table 5: Examples of actions and the corresponding action prompts for point-based tasks.

Action name	Action prompt
`move`	Move the point at index {index} by displacement {displacement}.
`move_towards`	Move the point at index {from_index} towards the point at index {to_index} by {distance}.
`insert_between`	Insert a new point between points at indices {index1} and {index2}, {distance} units away from point {index1}.
`rotate_around`	Rotate all points by {angle_deg} degrees around the axis {axis}, with the point at index {center_index} as the center of rotation. The rotation follows the right-hand rule.

A.2 Full Prompt Templates

Listing 1: A prompt example for a specific task of AtomWorld

```
You are a CIF operation assistant. You will be given an input CIF content
 and an action prompt. Your task is to apply the action described in the
action prompt to the initial CIF content. The coordinates in the action
are in Cartesian format. Return the modified CIF content in cif format
within <cif> and </cif> tags.

Please ensure the output is a valid CIF file, with correct formula, and
atom positions.

Input CIF content:
{The specific CIF file is inserted here}

Action prompt: Insert Lu between atoms at indices 6 and 5 that is 4.03
angstrom from atom 6.
```

Listing 2: A prompt example for the PointWorld task

```
You are a spatial reasoning expert. You will be given an initial set of
points and an action prompt describing an operation on these points. The
final modified points after applying the action must be returned inside <
answer> and </answer> tags. The format inside the tags must exactly match
 the input points format. All indices are zero-based. Please ensure the
answer inside <answer> and </answer> tags is parseable and strictly
formatted.
Initial points data:
{coordinate_array},
Action prompt:
{action_prompt},
```

Listing 3: A prompt example for CIF-repair tasks

```
You are a CIF operation assistant. You will be given a CIF content that
may be corrupted or incomplete. Your task is to examine the CIF content
and fix any issues to ensure it is a valid CIF file. If there are missing
 values that cannot be repaired directly, you can use the [
VALUE_TO_BE_INSERTED] as hints to fill in the missing values. Please
ensure the output is a correct CIF file. Return the fixed CIF content
within <cif> and </cif> tags. Input CIF content:

{broken_cif}
```

Listing 4: A prompt example for a CIF-gen task about perovskite structure

```
You are a materials science expert. Please generate some simple and
standard structures in the CIF format according to the requirements. You
must strictly follow the CIF format specifications. Since the symmetry-
related information can be complex, please write the CIF file with P1
symmetry. Please ensure the output is a correct CIF file. Return the
fixed CIF content within <cif> and </cif> tags.

Requirements:

Please generate a CIF file for {formula} with a {structure_type}
structure, according to the following information about the convensional
cell:

- Lattice constant a: {lattice_constant_a}
- The {center_atom} atom is at the center of the octahedron formed by
surrounding atoms.
```

Listing 5: A prompt example for StructProp tasks

```
You are a material design expert. Your task is to modify a given CIF file
 to achieve a desired change in a specific material property. Please
analyze the given CIF file and the target property. Identify the key
structural features and elemental composition that influence the
specified property. Propose a specific modification to the structure.
This modification must be one or a combination of the following:
 1. Element Substitution;
 2. Lattice Parameter Adjustment;
 3. Atomic Coordinate Adjustment.
 Please ensure the output is a correct CIF file. Return the modified CIF
 content within <cif> and </cif> tags.
 Input CIF content:
 {The specific CIF file is inserted here}

 Your goal: modify the CIF file accordingly to {target_trend} the {
 target_property}.
```

A.3 Illustrative example of the framework

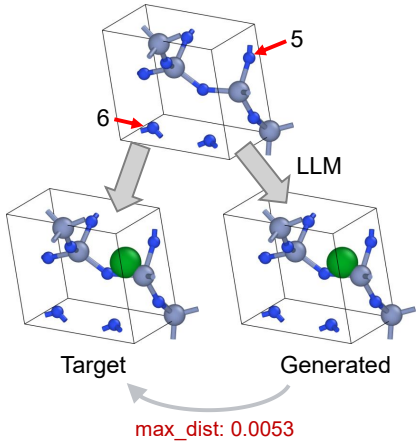

Figure 4: The workflow of a specific `insert_between` task.

To provide a concrete understanding of our proposed AtomWorld Bench, we present an illustrative example of its workflow. This case study focuses on a specific task: inserting a Lu atom between the fifth and the sixth atoms in the specific CIF structure. The prompt used here is listed in Appendix A.2.

The workflow randomly selects the atom indices and determines the position of the atom to be inserted based on the selected atoms. Based on the initialized action, the framework gives out a target structure. The LLM will also generate a structure after processing the prompt, as shown in Figure 4. In this example, the two structures are nearly identical, with a `max_dist` of 0.0053 Å, indicating high accuracy.

A.4 Logic on generating CIF-Repair task

To systematically evaluate LLM performance on CIF repair, we constructed a set of partially corrupted CIFs via two types of operations:

1. **Removal of essential lines:** Certain CIF fields are critical for correct structure parsing. The essential tags include:
   - `_cell_length_a`, `_cell_length_b`, `_cell_length_c`
   - `_cell_angle_alpha`, `_cell_angle_beta`, `_cell_angle_gamma`
   - `_atom_site_type_symbol`, `_atom_site_label`, `_atom_site_symmetry_multiplicity`
   - `_atom_site_fract_x`, `_atom_site_fract_y`, `_atom_site_fract_z`
   - `_atom_site_occupancy`
2. **Replacement of essential tags with misleading variants:** Instead of random typos, tags are systematically replaced with misleading but syntactically valid alternatives. Examples of mappings include:
   - Change the `a, b, c` into `x, y, z`; `u, v, w` or `i, j, k`.
   - Change the `x, y, z` into `a, b, c`; `u, v, w` or `i, j, k`.
   - Change `_atom_site` string into `_atom`.
   - Change `_cell` string into `_lattice`.
   - Change `_cell_length` and `_cell_angle` strings into `_cell`.

A.5 DFT computation details

All density functional theory (DFT) calculations, including band gap and bulk modulus evaluations, were performed using the Vienna Ab initio Simulation Package (VASP) with the projector-augmented wave (PAW) method (Kresse & Hafner, 1993; Kresse & Furthmüller, 1996a;b; Kresse & Joubert, 1999) and the PBEsol exchange–correlation functional(Perdew et al., 2008). High-throughput workflows for both properties were automated using the atomate2 package (Ganose et al., 2025). Unless otherwise specified, calculation parameters followed the default settings in atomate2. Example calculation scripts are provided in the github repository.

For the band gap calculations, a k-point mesh with a grid density of 100 Å^{-3} was employed, and electronic self-consistency was converged to 10^{-5} eV. The band gap was extracted from the uniform k-point calculation stage. For the bulk modulus calculations, a plane-wave energy cutoff of 600 eV and a k-point grid density of 400 Å^{-3} were used. Total energy and ionic relaxations were converged to 10^{-6} eV and 0.01 eV/Å, respectively, to balance computational cost and accuracy. In the initial relaxation stage, Gaussian smearing with σ = 0.05 eV was applied, while in the deformation stage the tetrahedron method was adopted for Brillouin zone integration.

B Full Evaluation Results

B.1 Evaluations of tool augmented LLM for AtomWorld

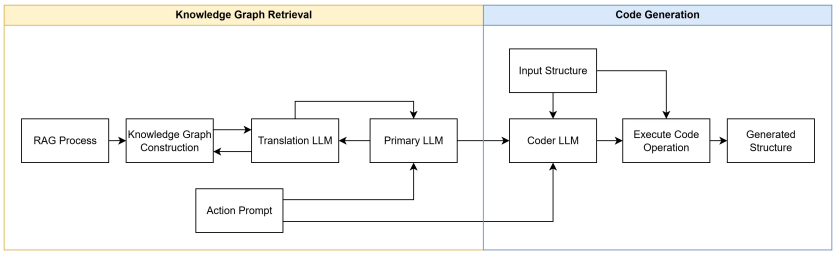

Figure 5: The flowchart for the code generation-based approach for the AtomWorld benchmark tests.

System Design As shown in Figure 5, we adopt a code generation-based approach to accomplish structural operations. This process is divided into two steps: first, we perform RAG-based retrieval over the pymatgen library to obtain relevant APIs; second, we conduct code generation to complete the user-specified action.

Knowledge Graph Retrieval (RAG) The first step of our pipeline is to retrieve relevant pymatgen APIs using RAG. We leverage the code-graph-rag project(Liu et al., 2024) to extract structured information from the codebase and build a knowledge graph in Memgraph, where nodes represent code entities such as modules, classes, methods, and fields, and edges capture relationships like inheritance and usage. The retrieval process is orchestrated by a primary LLM, implemented using Deepseek-chat, which performs task decomposition, reasoning, and tool invocation. Specifically, the translator LLM, also implemented with Deepseek-chat, is used as a tool by the primary LLM to convert natural language queries into graph queries. The output of this process is a JSON file containing relevant pymatgen APIs, which is later used to guide code generation.

Code Generation Code generation is performed using Deepseek-chat, conditioned on the input CIF file, the user action prompt, and the APIs retrieved from the RAG stage. The system strictly follows the retrieved API signatures to ensure correctness and prevent hallucination. The generated Python code is then executed together with the input CIF file to produce the modified crystal structure.

Table 6: Comparison of model performances between Deepseek-chat with and without tools.

	With tools		Without tools	
Action	Succ. rate (%)	mean `max_dist` (Å)	Succ. rate	mean `max_dist`
`remove`	**100.0**	0.0000	84.0	0.0000
`insert_between`	**83.0**	0.0076	45.6	0.2004
`rotate_around`	**18.0**	0.1648	6.8	0.2561

As evident from Table 6, incorporating retrieval-augmented generation (RAG) and structure manipulation tools significantly improves the model's performance across the tested actions. The `remove` action, which is relatively straightforward, achieves a perfect success rate of 100%. However, more complex actions, such as `insert_between` and `rotate_around`, still present challenges. The success rate for `insert_between` is 83%, with some errors remaining, while `rotate_around` demonstrates a relatively low success rate of 18%.

These findings highlight a key insight: while the integration of RAG tools and coding ability facilitates substantial improvements in model performance, further refinements are crucial to fully address the real-world requirements of structural modification tasks. Specifically, additional task-specific fine-tuning or reinforcement learning is necessary to enhance the model's robustness, particularly for

more complex structural operations. Future work will focus on these aspects to ensure more reliable and scalable applications.

B.2 THE MAX_DIST VIOLIN PLOTS

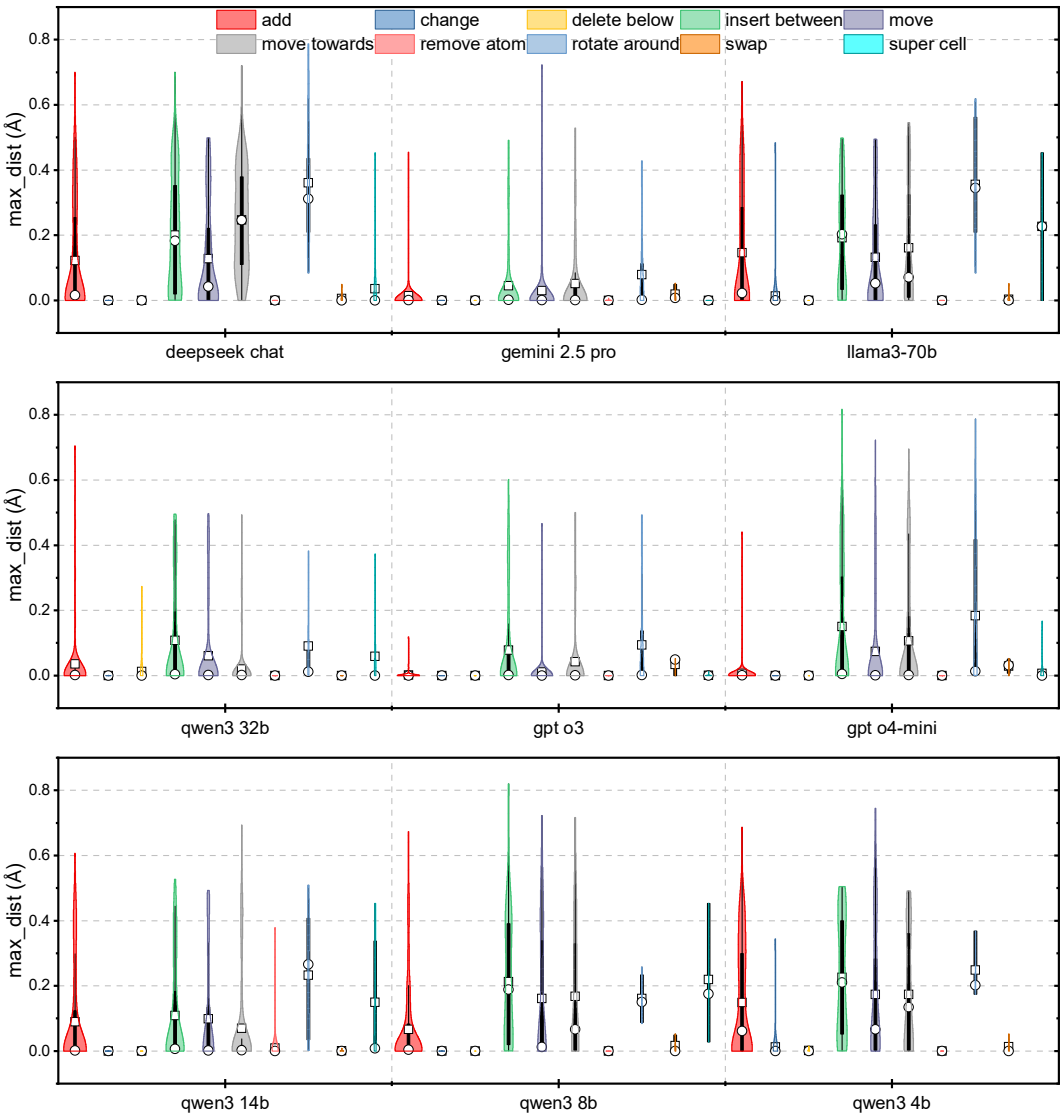

Figure 6: The violin plots of `max_dist` of evaluation results. The hollow squares indicate the mean values, and the hollow circles indicate the medians.