MT-MOL: Multi Agent System with Tool-based Reasoning for Molecular Optimization

Anonymous ACL submission

Abstract

Large language models (LLMs) have large potential for molecular optimization, as they can gather external chemistry tools and enable collaborative interactions to iteratively refine molecular candidates. However, this potential remains underexplored, particularly in the context of structured reasoning, interpretability, and comprehensive tool-grounded molecular optimization. To address this gap, we introduce MT-MOL, a multi-agent framework for molecular optimization that leverages tool-guided reasoning and role-specialized LLM agents. Our system incorporates comprehensive RDKit tools, categorized into five distinct domains: structural descriptors, electronic and topological features, fragment-based functional groups, molecular representations, and miscellaneous chemical properties. Each category is managed by an expert analyst agent, responsible for extracting task-relevant tools and enabling interpretable, chemically grounded feedback. MT-MOL produces molecules with tool-aligned and stepwise reasoning through the interaction between the analyst agents, a molecule-generating scientist, a reasoning-output verifier, and a reviewer agent. As a result, we show that our framework shows the state-of-the-art performance of the PMO-1K benchmark on 17 out of 23 tasks.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities in a wide range of problems such as question answering (Dong et al., 2024; Sun et al., 2024), summarization (Kim et al., 2024; Liu et al.), translation (Alves et al., 2024; Bari et al., 2025), and code generation (Chen et al., 2021; Li et al., 2023b) using large-scale pretraining and in-context learning (Brown et al., 2020; Chowdhery et al., 2023; Zhang et al., 2022b) (Chen et al., 2021). Motivated by the success, researchers are investigating the potential of LLMs for scientific discovery in the chemical domain (Wang et al., 2023; Luu et al., 2021; Wang et al., 2025; Nguyen and Grover, 2025; Bran et al., 2024).

In particular, employing LLMs to design new molecules (e.g., drug candidates), is promising due to several advantages: (1) LLMs exhibit general understanding and reasoning capabilities obtained from large-scale pretraining, (2) they can use the off-the-shelf tools for analyzing molecules, and (3) they are capable of interact with other agents to further improve the design candidate.

Recent studies have explored the application of LLMs in molecular optimization. For example, LICO (Nguyen and Grover, 2025) extends LLMs with embedding layers and in-context examples to build a surrogate modeling framework for molecular optimization. MOLLEO (Wang et al., 2025) leverages LLMs as mutation and crossover operators within an evolutionary algorithm. ChemCrow (Bran et al., 2024) integrates LLMs with chemical tools for to faciliate synthesis planning and molecular analysis. While these approaches demonstrate encouraging results, we argue that they do not fully exploit the broader capabilities of modern LLMs such as multi-agent collaboration, tool integration, and iterative reasoning, which are essential for high-quality molecular optimization.

Contribution. In this work, we propose MT-MOL, a multi-agent framework for molecular optimization. Our key idea is to decompose the optimization process into four distinct roles (**analyst**, **scientist**, **verifier**, and **reviewer**) and employ specialized agent for each role. Unlike previous approaches, MT-MOL generates molecules with explicit stepwise reasoning, consistency checks, and tool-informed feedback. Furthermore, we collect a set of 154 chemistry-related functions, which serve as applicable tools for agents during molecular generation process.

To be specific, we introduce four agents: (1) an-

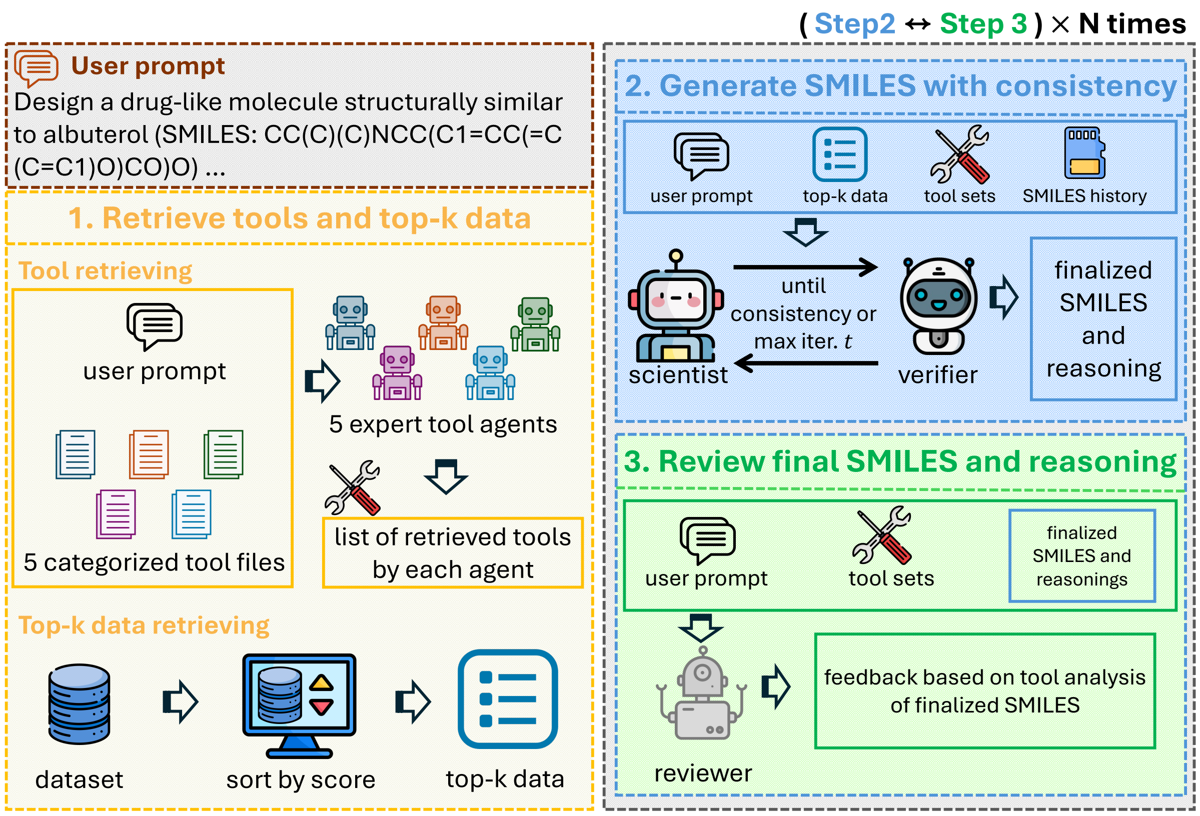

(Step2 ↔ Step 3) × N times
User prompt
Design a drug-like molecule structurally similar to albuterol (SMILES: CC(C)(C)NCC(C1=CC(=C(C=C1)O)CO)O) ...
1. Retrieve tools and top-k data
Tool retrieving
user prompt
5 categorized tool files
5 expert tool agents
list of retrieved tools by each agent
Top-k data retrieving
dataset
sort by score
top-k data
2. Generate SMILES with consistency
user prompt
top-k data
tool sets
SMILES history
scientist
until consistency or max iter. t
verifier
finalized SMILES and reasoning
3. Review final SMILES and reasoning
user prompt
tool sets
finalized SMILES and reasonings
reviewer
feedback based on tool analysis of finalized SMILES

Figure 1: **Overview of our method.** Given a molecular optimization task, analyst agents analyze the prompt and outputs list of relevant RDKit functions from five categories. Top-k molecules are retrieved as reference molecules for the scientist agent. Then, the scientist agent proposes a SMILES with stepwise reasoning, which the double checker validates for consistency. The reviewer finally assesses the reasoning using tool-informed descriptors and provides structured feedback. This generation and review process is repeated until the maximum number of iterations N is reached. This multi-agent pipeline enables interpretable, tool-guided molecule generation with iterative refinement toward the design objective.

alyst, (2) scientist, (3) verifier, and (4) reviewer. In detail, five **analyst agents** proposes the task-specific relevant tools using different types of chemical functions: structure, electronic properties, functional groups, identifiers, and miscellaneous descriptors. Then a **scientist agent** proposes new molecules in SMILES format (Weininger, 1988) and explains each design step through structured reasoning. Next, a **verifier agent** evaluates whether the reasoning of the scientist is consistent with the proposed molecule. Finally, a **reviewer agent** assesses both the molecule and the reasoning process using the outputs from the tools and provides detailed feedback. Each agent plays a collaborative role that enables interpretable, tool-aware, and iterative molecular design. By incorporating domain-specific tools such as RDKit (Landrum, 2013), MT-MOL supports chemically informed generation and transparent decision-making.

In summary, we propose a multi-agent framework for molecular optimization, where each agent is assigned a specific role such as tool selection, molecule generation, consistency validation, and reasoning critique. Our system integrates 154 RDKit functions, organized into five specialized analyst agents covering structural descriptors, electronic and topological descriptors, structural descriptors, fragment-based analysis, and identifiers or representations. We achieve state-of-the-art performance on 17 out of 23 tasks from the PMO-1K benchmark, outperforming recent strong baselines including LICO and MOLLEO in terms of top-10 AUC scores. Additionally, our framework offers an interpretable reasoning pipeline in which each generated molecule is equipped with stepwise rationale, double-check verification, and tool-informed reviewer feedback.

2 Related Work

Generative models for molecular optimization. Molecular optimization aims to design molecules that maximize desired chemical or biological properties, such as solubility, binding affinity, or synthesizability. Generative modeling has emerged as a central approach for this task, encompassing techniques from deep learning to probabilistic search. REINVENT (Olivecrona et al., 2017) introduced reinforcement learning over SMILES strings to fine-tune molecular generation toward desired properties. Jensen (2019) showed that graph-based genetic algorithms and non-ML models combined with Monte Carlo Tree Search perform competitively in optimizing molecular properties under synthetic constraints. Augmented Memory (Guo and Schwaller, 2024) enhances sample efficiency in reinforcement learning through SMILES augmentation and experience replay. Genetic GFN (Bengio et al., 2023) enables compositional molecule generation by sampling in proportion to a reward function, offering diversity and high-reward sampling in molecular benchmarks. Srinivas et al. (2010) introduced GP BO, a Gaussian process-based optimization framework that provides sublinear regret bounds and sample-efficient exploration using information gain from kernel-based uncertainty modeling. While these models improve sample efficiency and diversity, they often lack interpretability and fail to fully utilize the available domain knowledge, such as chemical priors. Our framework complements these approaches by incorporating structured reasoning and chemical tools into the molecular generation process.

LLMs for molecular optimization. LLMs have recently been applied to molecular optimization tasks. LICO (Nguyen and Grover, 2025) extends a pretrained LLM with structured embeddings to model property functions without relying on natural language prompts. MOLLEO (Wang et al., 2025) uses LLMs as evolutionary operators, enabling coherent molecule generation across single- and multi-objective settings. Prompt-MolOpt (Wu et al., 2024) introduces prompt-based editing to optimize multiple properties in low-data regimes while preserving pharmacophores. DrugAssist (Ye et al., 2025) fine-tunes an instruction-based LLM on a curated chemistry dataset to support interactive, feedback-driven molecule design. ChemCrow (Bran et al., 2024) combines general-purpose LLMs with chemistry tools and a ReAct-based reasoning loop to automate generation, retrosynthesis, and property prediction. Despite these advances, existing approaches often lack interpretability, structured collaboration among specialized agents, and a systematic feedback loop that enhances accurate molecule design. To address these limitations, our method introduces five expert analyst agents powered by RDKit (Landrum, 2013) and a multi-agent feedback loop that ensures both accurate and interpretable molecular optimization.

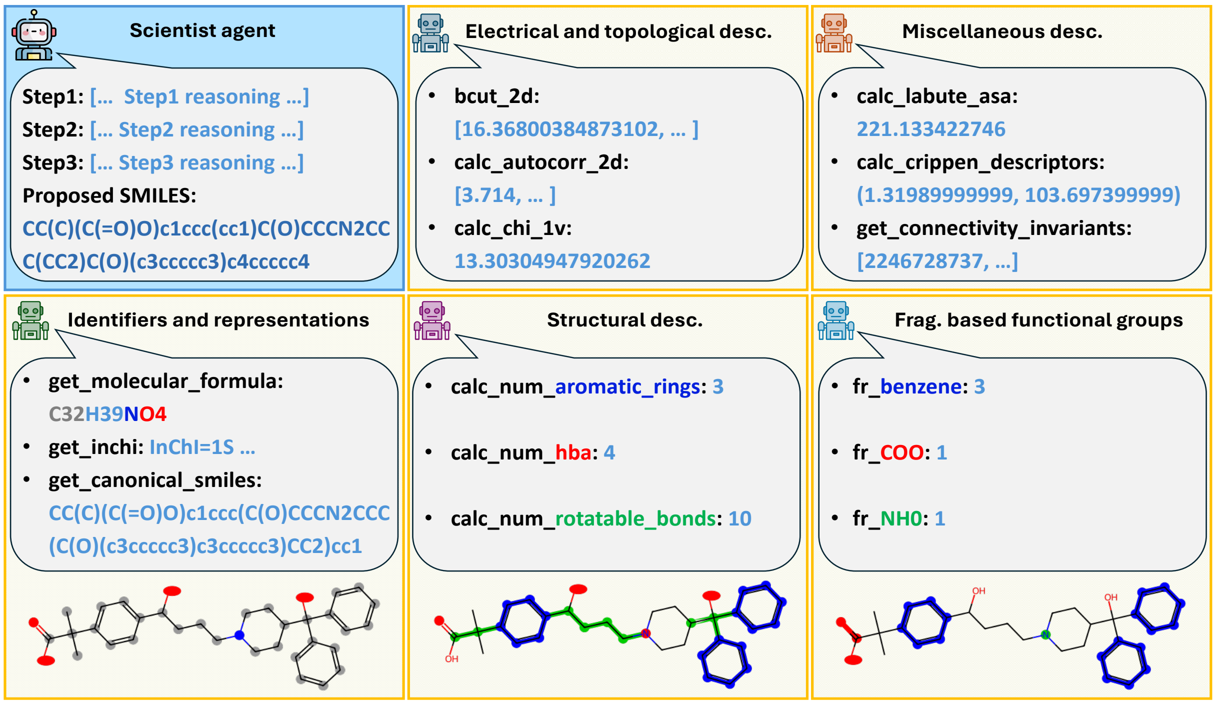
Multi-agent LLMs. Multi-agent LLMs have shown promise in collaborative reasoning and decomposed problem-solving. AgentVerse (Chen et al., 2023) assigns agents to roles like recruitment and evaluation, leveraging specialization for better coordination. ProAgent (Zhang et al., 2024) enables agents to infer and adapt to teammates' strategies through communication history. Self-Adaptive Multi-agent Systems (Nascimento et al., 2023) use a self-control loop to make agents responsive to dynamic environments. Theory of Mind for Multi-Agent Collaboration (Li et al., 2023a) enhances coordination by giving agents shared belief states and goal-tracking abilities. MetaGPT (Hong et al., 2024) improves communication scalability via a Shared Message Pool that standardizes agent interactions. While these frameworks contribute to multi-agent architectural design, they have overlooked domain-specific tool integration and have less focused on molecule optimization. Our framework addresses this gap by tightly coupling expert analyst agents with reasoning roles to enable targeted, tool-informed molecular design.

3 Method

In this section, we introduce our multi-agent framework for molecular optimization, coined MT-Mol. In Section 3.1, we first describe the overview of our system, which consists of four primary agent types: 1) analyst, 2) scientist, 3) verifier, and 4) reviewer agents. We describe details of the analysts in Section 3.2, and stepwise reasoning and feedback process in Section 3.3.

3.1 Overall Framework

In this section, we present a high-level overview of our multi-agent framework for molecular optimization. Given a user prompt T, analyst agents first select relevant tools, then the scientist agent proposes a molecule with structured reasoning. The verifier agent then verifies the logical consistency of the proposed output. Finally, the reviewer agent provides detailed feedback grounded in chemical analysis tools. We provide an overview of our method in Figure 1.


Scientist agent
Electrical and topological desc.
Miscellaneous desc.
Identifiers and representations
Structural desc.
Frag. based functional groups
OH
OH
OH

Step1: [... Step1 reasoning ...]
Step2: [... Step2 reasoning ...]
Step3: [... Step3 reasoning ...]
Proposed SMILES:
CC(C)(C(=O)O)c1ccc(cc1)C(O)CCCN2CCC(CC2)C(O)(c3ccccc3)c4ccccc4

- bcut_2d: [16.36800384873102, ...]
- calc_autocorr_2d: [3.714, ...]
- calc_chi_1v: 13.30304947920262

- calc_labute_asa: 221.133422746
- calc_crippen_descriptors: (1.31989999999, 103.697399999)
- get_connectivity_invariants: [2246728737, ...]

- get_molecular_formula: C32H39NO4
- get_inchi: InChI=1S ...
- get_canonical_smiles: CC(C)(C(=O)O)c1ccc(C(O)CCCN2CCC(C(O)(c3ccccc3)c3ccccc3)CC2)cc1

- calc_num_aromatic_rings: 3
- calc_num_hba: 4
- calc_num_rotatable_bonds: 10

- fr_benzene: 3
- fr_COO: 1
- fr_NH0: 1

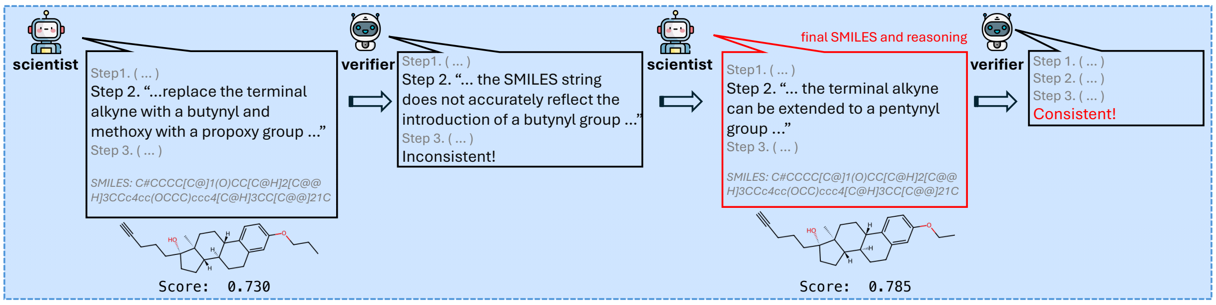
Figure 2: **Example of analyst agents.** Example case of five analyst agents analyzing the SMILES proposed by the scientist agent for the fexofenadine_mpo task. Each analyst agent chooses task-relevant tools: electronical and topological descriptors, miscellaneous descriptors, identifiers and representations, structural descriptors, and functional groups. The molecules at the bottom visualizes how analyst agents analyze the scientist agent's proposed SMILES. We provide the description of the tools at Appendix A.

Notably, our agents are informed about the details of the objective function and utilize their chemical knowledge to propose better molecules. This is in contrast to existing non-LLM works in molecular design that assume black-box objective functions. We believe that this is a strength of our approach, since in most of the tasks, we have some information about the objective function that can be described in natural language.

Analysts. We design five analyst agents for different aspects of molecular analysis. Each analyst agent parses and analyzes the molecule in the task prompt T and the scientist agent's proposed SMILES. Each analyst agent wraps a curated set of RDKit or PubChem functions in one of the following categories: 1) structural descriptors, 2) electronic and topological descriptors, 3) fragment-based functional group detectors, 4) chemical identifiers and representations, and 5) miscellaneous descriptors agents. To analyze a task prompt T, the analyst agents identify the most relevant chemical features and select the tools accordingly. We illustrate the example case of how tools are used in Figure 2 and provide the details of the tool in Appendix A.

Scientist. The scientist agent generates a novel molecule in SMILES format, denoted S, along with a reasoning path for proposing the molecule. To this end, the agent utilizes the tool-based analysis of the task prompt and a history of previously generated SMILES to avoid duplication. Based on the collected information, the agent proposes a molecule design strategy. It outlines this strategy in a sequence of k reasoning steps $\{r_1, \ldots, r_k\}$, where each r_i explains how the scientist agent thinks when proposing the SMILES representation of a molecule. After the reasoning process, the agent generates a SMILES string S.

Verifier. As noted by Pan et al. (2025), reasoning–action mismatch is a critical issue in multi-agent frameworks. To mitigate this in our system, we introduce a verifier agent that verifies each reasoning step in $\{r_1, \ldots, r_k\}$, ensuring that every r_i is faithfully reflected in the proposed SMILES S. In detail, it parses each step r_i and examines whether S contains the corresponding molecular feature. When discrepancies arise (e.g., when a reasoning step claims the presence of a nitro group, but S lacks it), the agent flags the inconsistency and produces stepwise feedbacks $\{f_1^v, \ldots, f_k^v\}$. Then, the verifier asks the scientist to re-generate the



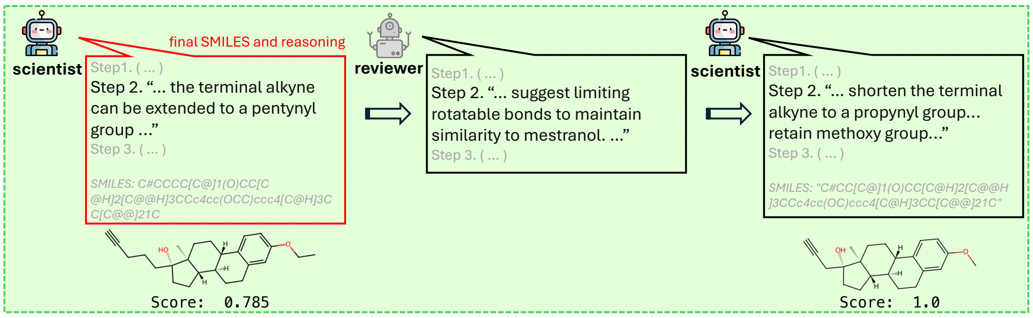
Step1. (...)
Step 2. "...replace the terminal alkyne with a butynyl and methoxy with a propoxy group ..."
Step 3. (...)
SMILES: C#CCCC[C@]1(O)CC[C@H]2[C@@H]3CCc4cc(OCCC)ccc4[C@H]3CC[C@@]21C

Step1. (...)
Step 2. "... the SMILES string does not accurately reflect the introduction of a butynyl group ..."
Step 3. (...)
Inconsistent!

Step1. (...)
Step 2. "... the terminal alkyne can be extended to a pentynyl group ..."
Step 3. (...)
SMILES: C#CCCC[C@]1(O)CC[C@H]2[C@@H]3CCc4cc(OCC)ccc4[C@H]3CC[C@@]21C

Step 1. (...)
Step 2. (...)
Step 3. (...)
Consistent!

(a) Structured feedback of the verifier agent.



Step1. (...)
Step 2. "... the terminal alkyne can be extended to a pentynyl group ..."
Step 3. (...)
SMILES: C#CCCC[C@]1(O)CC[C@H]2[C@@H]3CCc4cc(OCC)ccc4[C@H]3CC[C@@]21C

Step1. (...)
Step 2. "... suggest limiting rotatable bonds to maintain similarity to mestranol. ..."
Step 3. (...)

Step1. (...)
Step 2. "... shorten the terminal alkyne to a propynyl group... retain methoxy group..."
Step 3. (...)
SMILES: "C#CC[C@]1(O)CC[C@H]2[C@@H]3CCc4cc(OC)ccc4[C@H]3CC[C@@]21C"

(b) Structured feedback of the reviewer agent.

Figure 3: **Examples of structured and stepwise response.** The figures illustrate examples of structured feedback mechanisms employed by our agent system for the mestranol_similarity task. (a) The verifier flags a mismatch between reasoning and SMILES and the scientist revises both for consistency. (b) The reviewer suggests reducing rotatable bonds and the scientist reflects the design, improving the score.

SMILES based on the feedback. This re-generation loop continues until the verifier confirms consistency between the reasoning and SMILES, or until a maximum number of iterations t reached. If there is no discrepancy detected, it passes the verified reasoning steps $\{r_1, \ldots, r_k\}$ and SMILES S to the reviewer agent.

Reviewer. Inspired by previous works using LLMs as reviewers (Hosseini and Horbach, 2023; Zhang et al., 2022a), we introduce a chemical reviewer agent that evaluates and provides informative feedback. Specifically, the reviewer agent evaluates the verified SMILES S and reasoning steps $\{r_1, \ldots, r_k\}$. Using tool-based analysis of S, it provides chemically grounded, stepwise feedback $\{f_1^r, \ldots, f_k^r\}$ aligned with the structure of the reasoning. This feedback includes confirmations of correct reasoning, identification of wrong or missing claims, and suggestions for revision. The scientist agent then uses this feedback to refine both the reasoning and molecule S in the next iteration, enabling iterative improvement.

3.2 Details of analyst agents

We implement our multi-agent system with specialized LLM agents, with analyst agents playing a key role in analyzing molecules using domain-specific RDKit (Landrum, 2013) functions. These tools guide molecule generation by providing relevant descriptors to the scientist agent and support the reviewer with interpretable feedback. To enable comprehensive tool utilization and decomposed analysis, we categorize the analyst agents into five molecule-specialized aspects. Each agent targets a distinct aspect of molecular analysis and contributing to a chemically informed and interpretable design process. We provide a detailed list of tools that analyst agents take at Appendix A.

Electronic and topological descriptors. This agent analyzes how electrons are distributed in a molecule and how its atoms are connected, helping to assess properties such as reactivity and stability. It captures patterns that are important for determining whether a molecule is likely to behave well as a drug. As shown in Figure 2, this includes features such as charge distribution.

Fragment-based functional groups. This agent breaks molecules down into recognizable building blocks, such as rings or functional groups, such as acids or amines, which are commonly used in chemistry. These fragments are easy to interpret and often appear in the stepwise reasoning provided

Task	GP BO	REINVENT	LICO-*L*	Genetic GFN	Graph GA	Aug. Mem.	MOLLEO-*B*	MOLLEO-*D**	MT-Mol-*D**
albuterol_similarity	0.636	0.496	0.656	0.664	0.583	0.557	0.886	0.883	**0.998**
amlodipine_mpo	0.519	0.472	0.541	0.534	0.501	0.489	0.637	0.540	**0.647**
celecoxib_rediscovery	0.411	0.370	0.447	0.447	0.424	0.385	0.402	0.512	**0.867**
deco_hop	0.593	0.572	0.596	0.604	0.581	0.579	0.588	0.574	**0.842**
drd2	0.857	0.775	0.859	0.809	0.833	0.795	**0.910**	0.812	0.756
fexofenadine_mpo	0.707	0.650	0.700	0.682	0.666	0.679	0.674	0.680	**0.883**
gsk3b	0.611	0.589	0.617	**0.637**	0.523	0.539	0.397	0.496	0.308
isomers_c7h8n2o2	0.545	0.725	0.779	0.738	0.735	0.661	0.737	0.850	**0.986**
isomers_c9h10n2o2pf2cl	0.599	0.630	0.672	0.656	0.630	0.596	0.635	0.832	**0.914**
jnk3	0.346	0.315	0.336	**0.409**	0.301	0.294	0.186	0.342	0.125
median1	0.213	0.205	0.217	0.219	0.208	0.219	0.236	0.193	**0.321**
median2	0.203	0.188	0.193	0.204	0.181	0.184	0.191	0.197	**0.322**
mestranol_similarity	0.427	0.379	0.423	0.414	0.362	0.393	0.399	0.630	**0.996**
osimertinib_mpo	0.766	0.737	0.759	0.763	0.751	0.761	0.779	0.753	**0.796**
perindopril_mpo	0.458	0.404	0.473	0.462	0.435	0.422	**0.655**	0.422	0.542
qed	0.912	0.921	0.925	**0.928**	0.914	0.923	0.919	**0.928**	0.903
ranolazine_mpo	**0.701**	0.574	0.687	0.623	0.620	0.614	0.640	0.516	0.233
scaffold_hop	0.478	0.447	0.480	0.485	0.461	0.460	0.473	0.464	**0.646**
sitagliptin_mpo	0.232	0.261	0.315	0.227	0.229	0.245	0.193	**0.328**	0.067
thiothixene_rediscovery	0.351	0.311	0.343	0.377	0.322	0.336	0.416	0.478	**0.719**
troglitazone_rediscovery	0.313	0.246	0.292	0.277	0.267	0.262	0.302	0.387	**0.841**
valsartan_smarts	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
zaleplon_mpo	0.392	0.406	0.404	0.400	0.374	0.415	0.392	0.409	**0.625**
Sum of scores (↑)	11.27	10.68	11.71	11.56	10.90	10.81	11.65	12.23	**15.42**

Table 1: **Results of PMO-1K benchmark.** Tasks are assessed using AUC top-10 averaged by multiple runs. Results with (*) are evaluated from 3 independent runs while the others are assessed from 5 independent runs. We mark the best result in **bold** and the second-best in underline for each task.

by the scientist agent. Figure 2 shows how the agent highlights specific substructures, such as aromatic rings, that are captured, which is a key component of the task.

Identifiers and representations. This agent translates molecules into standardized formats such as canonicalized SMILES representation, molecular formulas, etc. Figure 2 illustrates how the functional group agent identifies chemically significant motifs such as benzene rings and carboxylic acids, which reflect specific fragment-level reasoning steps and enable chemically grounded feedback.

Structural descriptors. This agent captures basic geometric and physical features of a molecule, such as the number of atoms or bonds it contains. These properties influence how a molecule might behave in real-world conditions, including how it binds to targets or dissolves. As shown in Figure 2, this agent helps evaluate aspects like bond rotatability or ring complexity.

Miscellaneous descriptors. Miscellaneous descriptors agent provides additional analysis that complements the outputs of other agents. It captures properties that might be overlooked, such as molecular surface area, hybridization patterns, or structural irregularities, and helps ensure that the generated molecule is chemically reasonable. As shown in Figure 2, it offers supplementary evidence that strengthens the overall reasoning process.

3.3 Structured and stepwise response

In order to ensure that the agent responds to every desired component (e.g., stepwise reasoning, feedback, and SMILES), we guide the agent to output in JSON format using OpenAI API's function[1]. Specifically, the scientist agent generates stepwise reasoning and SMILES, while the verifier and reviewer agents produce stepwise feedback in a designated JSON format.

Also, for a valid and interpretable response, we guide the agents to output stepwise reasoning and feedback. Specifically, the scientist output stepwise reasoning $\{r_1, \ldots, r_k\}$ when proposing a SMILES S. Then, the verifier agent ensures the scientist agent's stepwise reasoning $\{r_1, \ldots, r_k\}$ is consistent with the output SMILES by providing the interpretable feedback $\{f_1^d, \ldots, f_k^d\}$. We visualize the example case in Figure 3a. The verifier agent identifies an inconsistency between the scientist's reasoning and the SMILES, since the butynyl group is not encoded.

In addition, the reviewer critiques the reasoning of the scientist agent with stepwise feedback $\{f_1^r, \ldots, f_k^r\}$. As illustrated in Figure 3b, the reviewer agent highlights the issue of increased rotatable bonds. This leads the scientist to revise the design by shortening the alkyne and restoring the methoxy group, which significantly improves the structural similarity score to 1.0. This shows

[1]https://platform.openai.com/docs/guides/structured-outputs?api-mode=chat

that our approach enables high alignment to target molecule, interpretability, and validity of the properties. We provide a detailed prompt and response example in appendix B.

4 Experiments

In this section, we evaluate the effectiveness of our multi-agent LLM system for molecular optimization in low-budget settings. We conduct experiments on the practical molecular optimization (PMO)-1K benchmark, which contains 23 chemically diverse optimization tasks, ranging from rediscovery and scaffold hopping to multi-property objectives. Our framework consists of expert analyst agents—each specialized in task decomposition, SMILES generation, verification, and tool-informed feedback—that collaborate to produce interpretable and high-quality molecular optimization. We compare our results against existing LLM-driven and evolutionary baselines, including LICO and MOLLEO, using various backbone models. We describe the dataset and baselines below, followed by the experimental setting described in Section 4.1. We then present the main benchmark results in Table 1 and provide analysis in Section 4.2.

Datasets. We evaluate on the Practical Molecular Optimization benchmark (Gao et al., 2022), which comprises 23 molecular optimization tasks. Each task defines a specific molecular property or structural constraint, such as rediscovery of known drugs (e.g., celecoxib, thiothixene), similarity to target scaffolds, or maximization of molecular property scores such as quantitative estimate of drug-likeness (QED) or logP. Following Gao et al. (2022), we assess performance using the top-10 area under the curve (AUC), which measures the average property score over oracle calls. Additionally considering Nguyen and Grover (2025), the evaluation is conducted for 1K oracle calls, simulating a budget-constrained discovery setting. We use the ZINC 250K (Sterling and Irwin, 2015) dataset to retrieve the top-100 reference molecules for the scientist agent's prompt. We summarize the entire tasks and their descriptions in Appendix C. All molecules are represented in the SMILES format and evaluated using predefined black-box scoring functions consistent with the PMO benchmark protocol.

Baselines. We compare our framework against six baselines: GP BO (Srinivas et al., 2010), REINVENT (Olivecrona et al., 2017), LICO (Nguyen and Grover, 2025), and two variants of MOLLEO (Wang et al., 2025) (MOLLEO-B, and MOLLEO-D). MOLLEO operates through LLM-guided mutation and crossover, using different base models (BioT5 (Pei et al., 2023) and DeepSeek-V3 (Liu et al., 2024). We evaluated two versions of our framework (Ours-D) using DeepSeek-V3 as a backbone for all the agent roles.

Task	Setting	AUC-Top10
osimertinib_mpo	MT-MOL	$\mathbf{0.796 \pm 0.005}$
	w/o Tool	0.694 ± 0.054
	w/o Reviewer	0.619 ± 0.140
	w/o Double checker	0.704 ± 0.017
albuterol_similarity	MT-MOL	$\mathbf{0.998 \pm 0.000}$
	w/o Tool	0.750 ± 0.021
	w/o Reviewer	0.991 ± 0.003
	w/o Double checker	0.996 ± 0.003
mestranol_similarity	MT-MOL	$\mathbf{0.996 \pm 0.001}$
	w/o Tool	0.831 ± 0.052
	w/o Reviewer	0.990 ± 0.002
	w/o Double checker	0.994 ± 0.002

Table 2: **Ablation study.** AUC-Top10 score under different agent removals for each task.

4.1 PMO Benchmark

Table 1 reports the performance of our framework and competing methods in all 23 PMO tasks. MT-MOL-D*, achieves the best performance in 17 of 23 tasks, significantly outperforming all baselines, including MOLLEO and LICO. In particular, MT-MOL surpasses the SOTA AUC sum of 12.23 (MOLLEO-D*) with a score of 15.42, marking a substantial improvement in the overall efficiency of optimization. The performance gap is particularly large on chemically complex tasks such as celecoxib_rediscovery and amlodipine_mpo, where MT-MOL-D* outperforms the previous best by more than 0.3 AUC points.

In Figure 4, we visualize the top-10 AUC curves for every 23 PMO tasks. MT-MOL consistently achieves faster and higher AUC trajectories compared to MOLLEO-D* across tasks such as albuterol_similarity, amlodipine_mpo, osimertinib_mpo, and troglitazon_rediscovery. These results suggest that our tool-aware reasoning, stepwise validation, and multi-agent feedback loop generate the desired molecule SMILES in the early stage while achieving high oracle value. The improvements are especially pronounced in the early stages of generation, indicating that Mol-Agent makes more efficient use of oracle calls.

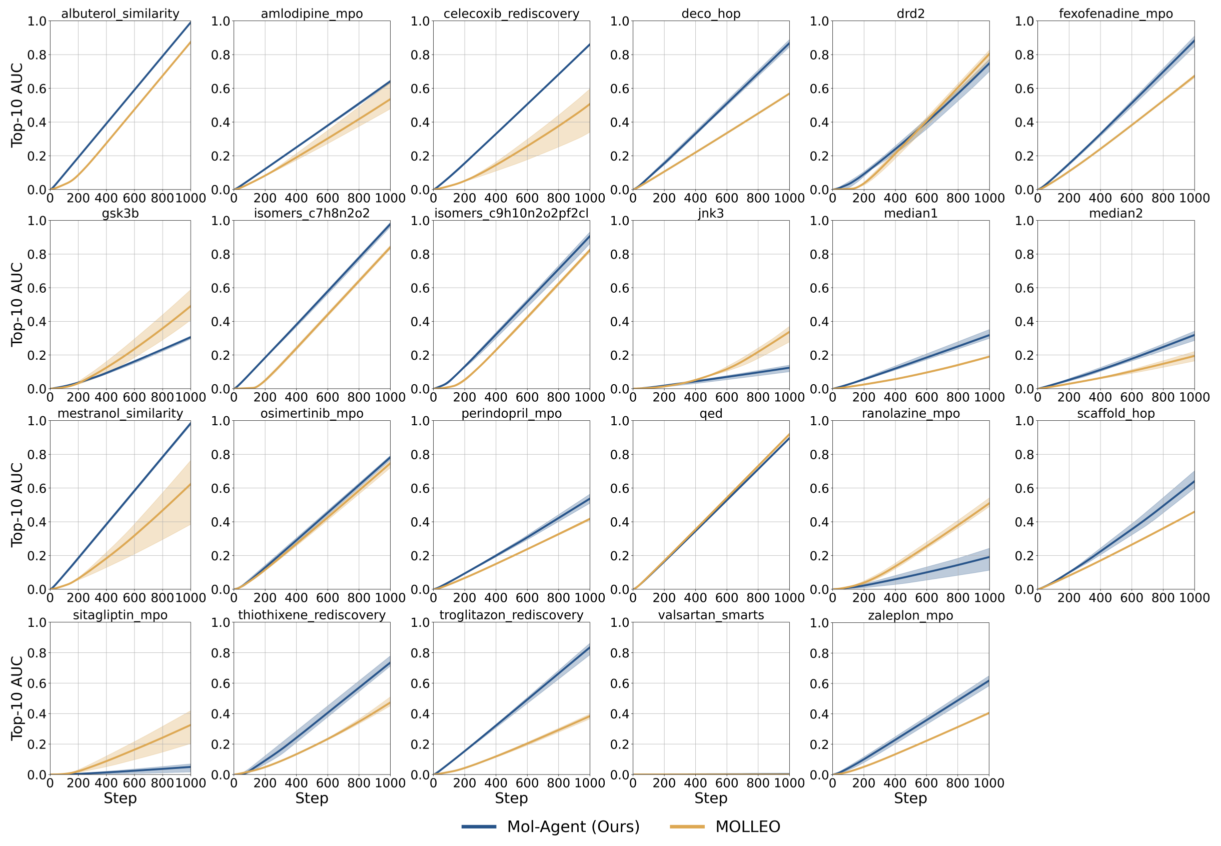


Figure 4: **Top-10 AUC curves.** Top-10 average AUC curves on the PMO benchmark, averaged over three random seeds. Our method consistently surpasses MOLLEO by achieving higher and faster-rising AUC curves, highlighting the effectiveness of tool-guided reasoning and multi-agent feedback in molecular optimization.

4.2 Ablation studies

To evaluate the contribution of each component in our multi-agent framework, we perform an ablation study on a subset of tasks from the PMO benchmark. Specifically, we assess the impact of removing (1) all five expert analyst agents, (2) the verifier agent, and (3) the reviewer agent. We report top-10 AUC scores averaged over three random seeds in Table 2.

One can observe that removing the analyst agents consistently leads to a substantial drop in performance across all tasks. For instance, the AUC score on albuterol_similarity drops from 0.998 to 0.750, highlighting that the expert analyst agents provide essential domain-specific descriptors.

Also removing reviwer agent causes noticeable degradation on tasks like osimertinib_mpo (from 0.796 to 0.619). Similarly, removing verifier aent shows modest performance drops when ablated, particularly on more challenging tasks.

Overall, these results underscore the importance of tool-guided analysis, structured reasoning verification, and feedback loops in our multi-agent system.

5 Conclusion

In this paper, we introduced MT-MOL, a multi-agent framework for molecular optimization and generation that combines tool-guided reasoning with structured collaboration among specialized LLM agents. Our system integrates five expert analyst agents, each equipped with domain-specific chemistry functions, to guide and critique molecule design. Through systematic interaction among scientist, verifier, and reviewer agents, MT-MOL achieves interpretable, chemically valid, and task-aligned molecular optimization. Our experiments on the PMO-1K benchmark demonstrate that MT-MOL outperforms strong baselines, including LICO and MOLLEO, achieving state-of-the-art performance on 17 out of 23 tasks. The results highlight the effectiveness of structured reasoning, tool-based validation, and multi-agent feedback in navigating the complex chemical space. This work provides the multi-agent system with comprehensive and systematic tool-augmented responses, accelerating molecular optimization and enabling transparent scientific discovery.

Broader Impact

Our work may help democratize access to molecular design expertise by enabling non-expert users to interact with intelligent agents that provide chemically grounded suggestions. Furthermore, the step-wise reasoning and feedback mechanisms embedded in our framework can serve as educational tools to help students and researchers understand the rationale behind molecule design decisions.

However, broader adoption of AI-assisted molecule design systems also raises potential ethical and social concerns. These include the misuse of generative tools for designing harmful substances, propagation of biases present in pretraining data, and the risk of over-reliance on AI-generated outputs without sufficient domain validation. Responsible deployment will require integrating safety checks, transparency mechanisms, and human-in-the-loop oversight.

Limitations

Our framework relies heavily on rule-based cheminformatics tools (e.g., RDKit) and predefined feature sets, which may limit generalization to novel chemical spaces or underrepresented functional groups. Moreover, while the multi-agent structure enables interpretability, it introduces additional computational overhead compared to single-agent models, potentially limiting scalability in resource-constrained settings.

Additionally, our experiments are conducted only in English and do not explore across other languages. This may limit usability in multilingual research environments or for integration with non-English scientific literature and databases.

References

Duarte M Alves, José Pombal, Nuno M Guerreiro, Pedro H Martins, João Alves, Amin Farajian, Ben Peters, Ricardo Rei, Patrick Fernandes, Sweta Agrawal, and 1 others. 2024. Tower: An open multilingual large language model for translation-related tasks. *Conference on Language Modeling*.

M Saiful Bari, Yazeed Alnumay, Norah A Alzahrani, Nouf M Alotaibi, Hisham A Alyahya, Sultan Al-Rashed, Faisal A Mirza, Shaykhah Z Alsubaie, Hassan A Alahmed, Ghadah Alabduljabbar, and 1 others. 2025. Allam: Large language models for arabic and english. *International Conference on Learning Representations*.

Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. 2023. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55.

Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. 2024. Chemcrow: Augmenting large-language models with chemistry tools. *Nature Machine Intelligence*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chen Qian, Chi-Min Chan, Yujia Qin, Yaxi Lu, Ruobing Xie, and 1 others. 2023. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents. *International Conference on Learning Representations*, 2(4):6.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Junnan Dong, Qinggang Zhang, Chuang Zhou, Hao Chen, Daochen Zha, and Xiao Huang. 2024. Cost-efficient knowledge-based question answering with large language models. *Advances in Neural Information Processing Systems*.

Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor Coley. 2022. Sample efficiency matters: a benchmark for practical molecular optimization. *Advances in neural information processing systems*, 35:21342–21357.

Jeff Guo and Philippe Schwaller. 2024. Augmented memory: Sample-efficient generative molecular design with reinforcement learning. *Jacs Au*, 4(6):2160–2172.

Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, and 1 others. 2024. Metagpt: Meta programming for multi-agent collaborative framework. *International Conference on Learning Representations*, 3(4):6.

Mohammad Hosseini and Serge PJM Horbach. 2023. Fighting reviewer fatigue or amplifying bias? considerations and recommendations for use of chatgpt and other large language models in scholarly peer review. *Research integrity and peer review*, 8(1):4.

Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohani, Jure Leskovec, Connor W Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. 2021. Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. *arXiv preprint arXiv:2102.09548*.

Jan H Jensen. 2019. A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space. *Chemical science*, 10(12):3567–3572.

Jaehyung Kim, Jaehyun Nam, Sangwoo Mo, Jongjin Park, Sang-Woo Lee, Minjoon Seo, Jung-Woo Ha, and Jinwoo Shin. 2024. Sure: Summarizing retrievals using answer candidates for open-domain qa of llms. *International Conference on Learning Representations*.

Greg Landrum. 2013. Rdkit documentation. *Release*, 1(1-79):4.

Huao Li, Yu Quan Chong, Simon Stepputtis, Joseph Campbell, Dana Hughes, Michael Lewis, and Katia Sycara. 2023a. Theory of mind for multi-agent collaboration via large language models. *Empirical Methods in Natural Language Processing*.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, and 1 others. 2023b. Starcoder: may the source be with you! *Transactions on Machine Learning Research*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Yixin Liu, AR Fabbri, P Liu, D Radev, and A Cohan. On learning to summarize with large language models as references (2024). *North American Chapter of the Association for Computational Linguistics*.

Kelvin Luu, Xinyi Wu, Rik Koncel-Kedziorski, Kyle Lo, Isabel Cachola, and Noah A Smith. 2021. Explaining relationships between scientific documents. ** Association for Computational Linguistics*.

Nathalia Nascimento, Paulo Alencar, and Donald Cowan. 2023. Self-adaptive large language model (llm)-based multiagent systems. In *2023 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*, pages 104–109. IEEE.

Tung Nguyen and Aditya Grover. 2025. Lico: Large language models for in-context molecular optimization. *International Conference on Learning Representations*.

Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. 2017. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9:1–14.

Melissa Z Pan, Mert Cemri, Lakshya A Agrawal, Shuyi Yang, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Kannan Ramchandran, Dan Klein, and 1 others. 2025. Why do multiagent systems fail? In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*.

Qizhi Pei, Wei Zhang, Jinhua Zhu, Kehan Wu, Kaiyuan Gao, Lijun Wu, Yingce Xia, and Rui Yan. 2023. Biot5: Enriching cross-modal integration in biology with chemical knowledge and natural language associations. *Empirical Methods in Natural Language Processing*.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. 2010. Gaussian process optimization in the bandit setting: No regret and experimental design. *International Conference on Machine Learning*.

Teague Sterling and John J Irwin. 2015. Zinc 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11):2324–2337.

Hongda Sun, Yuxuan Liu, Chengwei Wu, Haiyu Yan, Cheng Tai, Xin Gao, Shuo Shang, and Rui Yan. 2024. Harnessing multi-role capabilities of large language models for open-domain question answering. In *Proceedings of the ACM Web Conference 2024*, pages 4372–4382.

Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, and 1 others. 2023. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60.

Haorui Wang, Marta Skreta, Cher-Tian Ser, Wenhao Gao, Lingkai Kong, Felix Strieth-Kalthoff, Chenru Duan, Yuchen Zhuang, Yue Yu, Yanqiao Zhu, and 1 others. 2025. Efficient evolutionary search over chemical space with large language models. *International Conference on Learning Representations*.

David Weininger. 1988. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36.

Zhenxing Wu, Odin Zhang, Xiaorui Wang, Li Fu, Huifeng Zhao, Jike Wang, Hongyan Du, Dejun Jiang, Yafeng Deng, Dongsheng Cao, and 1 others. 2024. Leveraging language model for advanced multiproperty molecular optimization via prompt engineering. *Nature Machine Intelligence*, pages 1–11.

Geyan Ye, Xibao Cai, Houtim Lai, Xing Wang, Junhong Huang, Longyue Wang, Wei Liu, and Xiangxiang Zeng. 2025. Drugassist: A large language model for molecule optimization. *Briefings in Bioinformatics*, 26(1):bbae693.

Ceyao Zhang, Kaijie Yang, Siyi Hu, Zihao Wang, Guanghe Li, Yihang Sun, Cheng Zhang, Zhaowei Zhang, Anji Liu, Song-Chun Zhu, and 1 others. 2024. Proagent: building proactive cooperative agents with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17591–17599.

Jiayao Zhang, Hongming Zhang, Zhun Deng, and Dan Roth. 2022a. Investigating fairness disparities in peer review: A language model enhanced approach. *arXiv preprint arXiv:2211.06398*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, and 1 others. 2022b. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

A List of tools

List of tools are provided by categories. Tools of electronic and topological descriptors are provided at Table 3, fragment based functional groups at Table 4, Table 5, and Table 6, identifiers and representations at Table 7, structural descriptors at Table 9, and miscellaneous descriptors at Table 8.

Function	**Description**
`bcut2d`	Implements BCUT descriptors From J. Chem. Inf. Comput. Sci., Vol. 39, No. 1, 1999Diagonal elements are (currently) atomic mass, gasteiger charge,crippen logP and crippen MR-Returns the 2D BCUT2D descriptors vector as described in returns [mass eigen value high, mass eigen value low, gasteiger charge eigenvalue high, gasteiger charge low,
`calcautocorr2d`	Returns 2D Autocorrelation descriptor vector using a specified atom property.
`calcchi0n`	Calculates the Chi0n index, a valence-based topological descriptor.
`calcchi0v`	Calculates the Chi0v index, a non-valence-based topological descriptor.
`calcchi1n`	Calculates the Chi1n index using atom connectivity and optionally forces calculation.
`calcchi1v`	Calculates the Chi1v index, a valence-corrected form of Chi1n.
`calcchi2n`	Calculates the Chi2n index, a higher-order topological descriptor (non-valence- based).
`calcchi2v`	Calculates the Chi2n index, a higher-order topological descriptor (non-valence- based).
`calcchi3n`	Calculates the Chi3n index for extended connectivity (non-valence-based).
`calcchi3v`	Calculates the Chi3v index with valence correction for deeper molecular topology.
`calcchi4n`	Calculates the Chi4n index, further extending non-valence connectivity descriptors.
`calcchi4v`	Calculates the Chi4v index, a valence-aware descriptor at a 4th topological level

Table 3: List of Electronic Topological Descriptors tools

Function	Description
get_min_ring_frequency	Return the least frequent known ring system in the molecule with its frequency.
remove_stereo_from_smiles	Removes stereochemistry from SMILES and returns canonical SMILES and InChI Key.
get_spiro_atoms	Returns atom indices that are shared between two rings (spiro atoms).
max_ring_size	Returns the size of the largest ring in the molecule.
ring_stats	Returns the number of rings and the size of the largest ring in the molecule.
count_fragments	Returns the number of molecular fragments present in the SMILES.
get_largest_fragment	Returns the SMILES of the largest fragment by atom count in a molecule.
fr_phos_acid	Number of phosphoric acid groups
fr_Al_COO	Number of aliphatic carboxylic acids
fr_Al_OH	Number of aliphatic hydroxyl groups
fr_Al_OH_noTert	Number of aliphatic hydroxyl groups excluding tert-OH
fr_ArN	Number of N functional groups attached to aromatics
fr_Ar_COO	Number of Aromatic carboxylic acid
fr_Ar_N	Number of aromatic nitrogens
fr_Ar_NH	Number of aromatic amines
fr_Ar_OH	Number of aromatic hydroxyl groups
fr_COO	Number of carboxylic acids
fr_COO2	Number of carboxylic acids
fr_C_O	Number of carbonyl O
fr_C_O_noCOO	Number of carbonyl O, excluding COOH
fr_C_S	Number of thiocarbonyl
fr_HOCCN	Number of C(OH)CCN-Ctert-alkyl or C(OH)CCNcyclic
fr_Imine	Number of Imines
fr_NH0	Number of Tertiary amines
fr_NH1	Number of Secondary amines
fr_NH2	Number of Primary amines
fr_N_O	Number of hydroxylamine groups
fr_Nhpyrrole	Number of H-pyrrole nitrogens
fr_SH	Number of thiol groups
fr_aldehyde	Number of aldehydes
fr_alkyl_carbamate	Number of alkyl carbamates (subject to hydrolysis)
fr_alkyl_halide	Number of alkyl halides
fr_allylic_oxid	Number of allylic oxidation sites excluding steroid dienone
fr_amide	Number of amides
fr_amidine	Number of amidine groups
fr_aniline	Number of anilines
fr_aryl_methyl	Number of aryl methyl sites for hydroxylation
fr_azide	Number of azide groups
fr_azo	Number of azo groups
fr_barbitur	Number of barbiturate groups
fr_benzene	Number of benzene rings
fr_benzodiazepine	Number of benzodiazepines with no additional fused rings
fr_bicyclic	Bicyclic

Table 4: List of Fragment Based Functional Groups tools (1/3)

Function	**Description**
`fr_diazo`	Number of diazo groups
`fr_dihydropyridine`	Number of dihydropyridines
`fr_epoxide`	Number of epoxide rings
`fr_ester`	Number of esters
`fr_ether`	Number of ether oxygens (including phenoxy)
`fr_furan`	Number of furan rings
`fr_guanido`	Number of guanidine groups
`fr_halogen`	Number of halogens
`fr_hdrzine`	Number of hydrazine groups
`fr_hdrzone`	Number of hydrazone groups
`fr_imidazole`	Number of imidazole rings
`fr_imide`	Number of imide groups
`fr_isocyan`	Number of isocyanates
`fr_isothiocyan`	Number of isothiocyanates
`fr_ketone`	Number of ketones
`fr_ketone_Topliss`	Number of ketones excluding diaryl, a,b-unsat. dienones, heteroatom on Calpha

Table 5: List of Fragment Based Functional Groups tools (2/3)

Function	**Description**
`fr_lactam`	Number of beta lactams
`fr_lactone`	Number of cyclic esters (lactones)
`fr_methoxy`	Number of methoxy groups -OCH3
`fr_morpholine`	Number of morpholine rings
`fr_nitrile`	Number of nitriles
`fr_nitro`	Number of nitro groups
`fr_nitro_arom`	Number of nitro benzene ring substituents
`fr_nitro_arom_nonortho`	Number of non-ortho nitro benzene ring substituents
`fr_nitroso`	Number of nitroso groups, excluding NO2
`fr_oxazole`	Number of oxazole rings
`fr_oxime`	Number of oxime groups
`fr_para_hydroxylation`	Number of para-hydroxylation sites
`fr_phenol`	Number of phenols
`fr_phenol_noOrthoHbond`	Number of phenolic OH excluding ortho intramolecular Hbond substituents
`fr_phos_ester`	Number of phosphoric ester groups
`fr_piperdine`	Number of piperdine rings
`fr_piperzine`	Number of piperzine rings
`fr_priamide`	Number of primary amides
`fr_prisulfonamd`	Number of primary sulfonamides
`fr_pyridine`	Number of pyridine rings
`fr_quatN`	Number of quaternary nitrogens
`fr_sulfide`	Number of thioether
`fr_sulfonamd`	Number of sulfonamides
`fr_sulfone`	Number of sulfone groups
`fr_term_acetylene`	Number of terminal acetylenes
`fr_tetrazole`	Number of tetrazole rings
`fr_thiazole`	Number of thiazole rings
`fr_thiocyan`	Number of thiocyanates
`fr_thiophene`	Number of thiophene rings
`fr_unbrch_alkane`	Number of unbranched alkanes of at least 4 members (excludes halogenated alkanes)
`fr_urea`	Number of urea groups

Table 6: List of Fragment Based Functional Groups tools (3/3)

Function	**Description**
`get_rdkit_complexity`	Returns the Bertz molecular complexity index of the molecule.
`get_rdkit_number_of_atoms`	Returns the number of atoms in the molecule.
`get_rdkit_number_of_bonds`	Returns the number of bonds in the molecule.
`get_rdkit_rotatable_bond_count`	Returns the number of rotatable bonds in the molecule.
`get_rdkit_h_bond_donor_count`	Returns the number of hydrogen bond donors in the molecule.
`get_rdkit_h_bond_acceptor_count`	Returns the number of hydrogen bond acceptors in the molecule.
`get_rdkit_molecular_formula`	Returns the molecular formula of the molecule.
`get_rdkit_canonical_smiles`	Returns the canonical SMILES of the molecule.
`get_rdkit_inchi`	Returns the InChI string of the molecule.

Table 7: List of Identifiers and Representations tools

Function	**Description**
`smi2mol_with_errors`	Attempts to parse SMILES and returns validation status with error/warning messages.
`calcmolformula`	Returns the molecule 2019s formula
`calccrippendescriptors`	Returns a 2-tuple with the Wildman-Crippen logp,mr values
`calcfractioncsp3`	Returns the fraction of C atoms that are SP3 hybridized
`calckappa1`	Calculates the first Kier shape index, reflecting molecular linearity based on atom and bond counts.
`calckappa2`	Computes the second Kier shape index, indicating molecular cyclicity and branching.
`calckappa3`	Computes the third Kier shape index, sensitive to molecular flexibility and complex ring structures.
`calclabuteasa`	Returns the Labute ASA value for a molecule
`calcpbf`	Returns the PBF (plane of best fit) descriptor
`calcphi`	Estimates the molecular flexibility index based on the number of rotatable bonds and ring structures.
`getconnectivityinvariants`	Returns connectivity invariants (ECFP-like) for a molecule.
`getfeatureinvariants`	Returns feature invariants (FCFP-like) for a molecule.
`mqns_`	Computes Molecular Quantum Numbers, a 42-dimensional vector of counts for various atom types, bonds, and topological features.
`peoe_vsa_`	Computes descriptors combining partial charges (Gasteiger) with van der Waals surface areas in defined bins.
`smr_vsa_`	Calculates descriptors combining molar refractivity contributions with surface areas in predefined ranges.
`slogp_vsa_`	Computes descriptors by combining atomic logP contributions (Wildman-Crippen) with van der Waals surface areas.

Table 8: List of Other Descriptors tools

Function	**Description**
`get_center`	Computes the geometric center of a conformer generated from the input SMILES.
`get_shape_moments`	Calculates NPR1 and NPR2 shape descriptors from a generated conformer.
`refine_conformers`	Refines 3D conformers based on energy and RMSD thresholds.
`get_conformer_energies`	Returns the energies of multiple conformers generated from the input molecule.
`calcnumaliphaticcarbocycles`	Returns the number of aliphatic (containing at least one non-aromatic bond) carbocycles for a molecule
`calcnumaliphaticheterocycles`	Returns the number of aliphatic (containing at least one non-aromatic bond) heterocycles for a molecule
`calcnumaliphaticrings`	Returns the number of aliphatic (containing at least one non-aromatic bond) rings for a molecule
`calcnumamidebonds`	Returns the number of amide bonds in a molecule
`calcnumaromaticcarbocycles`	Returns the number of aromatic carbocycles for a molecule
`calcnumaromaticheterocycles`	Returns the number of aromatic heterocycles for a molecule
`calcnumaromaticrings`	Returns the number of aromatic rings for a molecule
`calcnumatomstereocenters`	Returns the total number of atomic stereocenters (specified and unspecified)
`calcnumatoms`	Returns the total number of atoms for a molecule
`calcnumhba`	Returns the number of H-bond acceptors for a molecule
`calcnumhbd`	Returns the number of H-bond donors for a molecule
`calcnumheavyatoms`	Returns the number of heavy atoms for a molecule
`calcnumheteroatoms`	Returns the number of heteroatoms for a molecule
`calcnumheterocycles`	Returns the number of heterocycles for a molecule
`calcnumlipinskihba`	Returns the number of Lipinski H-bond acceptors for a molecule
`calcnumlipinskihbd`	Returns the number of Lipinski H-bond donors for a molecule
`calcnumrings`	Returns the number of rings for a molecule
`calcnumrotatablebonds`	Returns the number of rotatable bonds for a molecule. strict = NumRotatableBondsOptions.NonStrict - Simple rotatable bond definition.
`calcnumsaturatedcarbocycles`	Returns the number of saturated carbocycles for a molecule
`calcnumsaturatedheterocycles`	Returns the number of saturated heterocycles for a molecule
`calcnumsaturatedrings`	Returns the number of saturated rings for a molecule
`calcnumunspecifiedatomstereocenters`	Returns the number of unspecified atomic stereocenters
`calcoxidationnumbers`	Adds the oxidation number/state to the atoms of a molecule as property OxidationNumber on each atom. Use Pauling electronegativities. This is experimental code, still under development.

Table 9: List of Structural Descriptors tools

B Prompts

B.1 Prompt for analyst

You are a professional AI chemistry assistant specialized in resolving [category name] using RDKit tools.
Your job is to identify how to retrieve standardized molecular information such as CIDs, InChI, and canonical SMILES for downstream processing.

Follow this structured reasoning process step-by-step:

Step 1. Analyze the molecule design condition which is the goal of the task.
Step 2. Parse list of all valid SMILES strings mentioned anywhere in the user prompt and output them in the provided JSON format.
Step 3. Based on your chemical knowledge, explain why standardizing identifiers and resolving canonical formats might be important for this task.
- E.g., checking uniqueness, linking to external data, verifying molecular identity.
Step 4. Choose as many tools as necessary from the identifier toolset that help you access consistent molecular representations or external references.
Step 5. Output your final answer in the provided JSON format.

This is a molecule design condition of the [task name] task: [task description]
Now output the tools to use by using the following JSON format. Take a deep breath and think carefully before writing your answer. “‘json {{
"parsed_smiles": [
{{
"smiles": "Parsed SMILES string",
}} ,
...
],
"tools_to_use": [
{{
"tool_name": "fr_Ar_OH",
"purpose": "Detect aromatic hydroxyl groups, similar to those in albuterol."
}},
...
]
}}

B.2 Prompt for scientist

You are a skilled chemist.

Your task is to design a SMILES string for a molecule that satisfies the following condition: [task description]

Functional groups and molecule tool analysis results of task related molecules: [result of tool analysis]

You are provided with:
- Top-100 example molecules with high relevance to the task, listed below. You may use these as inspiration, but YOU MUST NOT COPY THEM EXACTLY.
- A list of previously generated SMILES, which YOU MUST NOT REPEAT.

Top-100 Relevant SMILES Examples (SMILES, score)
YOU MUST FAITHFULLY REFER TO THESE EXAMPLES WHEN DESIGNING YOUR MOLECULE. BUT DO NOT COPY THEM EXACTLY:

[top100 SMLIES]

You must return your response in the following json format. The text inside each key explains what kind of answer is expected — it is a guideline, not the answer.

DO NOT repeat the example text or instructions. Instead, write your own scientifically reasoned content based on the task.

Use the following format. Take a deep breath and think carefully before writing your answer.

“‘json
{{
"step1": "List of the target’s critical structural/property features (e.g., ’Target: phenyl ring, β-hydroxyamine, catechol-like substitution’). If property-based, specify requirements (e.g., "logP > 3: add hydrophobic groups").",
"step2": "Propose modifications or scaffolds to meet the condition (e.g., ’Replace catechol with 3-hydroxy-4-pyridone’).
Justify each change chemically (e.g., "Maintains H-bonding but improves metabolic stability").",
"step3": "Describe the full structure of your designed molecule in natural language before writing the SMILES. (e.g., "A tert-butyl group attached to the amine (–NH–$C(CH_3)_3$) to mimic target’s bulky substituent.")",
"smiles": "Your valid SMILES string here"
}}

B.3 Prompts for scientist with feedback

YOU MUST NOT REPEAT ANY OF THE PREVIOUSLY GENERATED SMILES: [smiles_history]
Task: Take [verifier/reviewer]'s feedback actively and design a SMILES string for a molecule that satisfies the condition:

Condition for molecule design:
[task description]

Functional groups and molecule tool analysis results of task related molecules:
[target functional groups]

Top-100 Relevant SMILES Examples (SMILES, score)
YOU MUST FAITHFULLY REFER TO THESE EXAMPLES WHEN DESIGNING YOUR MOLECULE. BUT DO NOT COPY THEM EXACTLY:
[topk smiles]

You will be provided with:
1. Previous SMILES string
2. Task score (0–1)
3. Detected functional groups in your previous molecule

— MOLECULE SMILES TO IMPROVE —
MOLECULE SMILES: [previous smiles]
- Task score: [score] (0–1)
- Functional groups detected:
[functional groups]

— YOUR PREVIOUS THOUGHT AND REVIEWER'S FEEDBACK —
Step1: List Key Features
Your previous thought process:
[scientist step1 reasoning]
Accordingly, reviewer's feedback is:
[verifier/reviewer step1 feedback]

Step2: Design Strategy:
Your previous thought process:
[scientist step2 think]
Accordingly, reviewer's feedback is:
[verifier/reviewer step2 feedback]

Step 3: Construct the Molecule: Your previous thought process:
[verifier/scientist step3 think]
Accordingly, reviewer's feedback is:
[verifier/reviewer step3 feedback]

Now based on your previous thoughts and the reviewer's feedback, you need to improve your design.
You must return your response in the following json format.
The text inside each key explains what kind of answer is expected — it is a guideline, not the answer.

DO NOT repeat the example text or instructions.
Instead, write your own scientifically reasoned content based on the task.

Use the following format.
Take a deep breath and think carefully before writing your answer.
“‘json
{{
"step1": "List of the target’s critical structural/property features (e.g., ’Target: phenyl ring, β-hydroxyamine, catechol-like substitution’). If property-based, specify requirements (e.g., "logP > 3: add hydrophobic groups").",
"step2": "Propose modifications or scaffolds to meet the condition (e.g., ’Replace catechol with 3-hydroxy-4-pyridone’). Justify each change chemically (e.g., "Maintains H-bonding but improves metabolic stability").",
"step3": "Describe the full structure of your designed molecule in natural language before writing the SMILES. (e.g., "A tert-butyl group attached to the amine (–NH–$C(CH_3)_3$) to mimic target’s bulky substituent.")",
"smiles": "Your valid SMILES string here"
}}
“‘

B.4 Prompts for verifier

You are a meticulous double-checker LLM. Your task is to verify whether each step of the scientist's reasoning is chemically valid and faithfully and logically reflected in the final SMILES string.
You will be given:
- A user prompt describing the target objective,
- The scientist's reasoning broken into Step1 through Step3,
- The SMILES string proposed by the scientist.

Evaluate each step independently, comparing the described logic to the molecular structure in the SMILES.
Provide a reasoning assessment for each step. === SCIENTIST'S TASK ===
If any step is inconsistent, mark "Consistency" as "Inconsistent" and provide specific suggestions for improvement.

[task description]

Functional groups and molecule tool analysis results of task related molecules:
[target functional groups]

=== SCIENTIST'S THINKING ===
Step1: [thinking['step1']]
Step2: [thinking['step2']]
Step3: [thinking['step3']]

=== SCIENTIST'S SMILES ===
- SMILES: [smiles]
- Detected functional groups and molecule tool analysis results:

[functional groups]
You must return your response in the following json format.
The text inside each key explains what kind of answer is expected — it is a guideline, not the answer.

DO NOT repeat the example text or instructions.
Instead, write your own scientifically reasoned content based on the task.

Use the following format.
Take a deep breath and think carefully before writing your answer.
‘‘‘json {{
"step1": "Your analysis of whether scientist's Step1 thinking is chemically valid and reflected in the SMILES.",
"step2": "Your analysis of whether scientist's Step2 thinking is chemically valid and reflected in the SMILES.",
"step3": "Your analysis of whether scientist's Step3 thinking is chemically valid and reflected in the SMILES.",
"consistency": "Consistent" or "Inconsistent",
}}
‘‘‘

B.5 Prompts for reviewer

You are a rigorous chemistry reviewer.
Evaluate the Scientist LLM's reasoning steps and final SMILES molecule for:
- Validity
- Chemical soundness
- Adherence to the design condition:

Scientist LLM's task:
[task description]

Be constructive: Provide fixes for issues (e.g., "Replace C=O=C with O=C=O for carbon dioxide").

You are provided with:
- Scientist's thinking
- Scientist-generated SMILES
- Task score
- Detected functional groups in the generated molecule

— SCIENTIST'S STEP-WISE THINKING —
Step 1: [scientist step1 reasoning]

Step 2: [scientist step2 reasoning]

Step 3: [scientist step3 reasoning]

— SCIENTIST-MOLECULE SMILES —
SMILES: [scientist proposed SMILES]
- Task score: [score] (range: 0 to 1)
- Detected functional groups and molecule tool analysis results:
[functional groups]

You must return your response in the following json format.
The text inside each key explains what kind of answer is expected — it is a guideline, not the answer.

DO NOT repeat the example text or instructions.
Instead, write your own scientifically reasoned content based on the task.

Use the following format.
Take a deep breath and think carefully before writing your answer.
```json
{{
"step1": "List accurate features and functional groups identified. Mention any critical features and functional groups that were missed or misinterpreted.",
"step2": "Evaluate if the proposed design strategy aligns with the structural and functional similarity goal.
Comment on whether the design aligns with the initial objectives. Suggest improvements or alternatives if needed.",

"step3": "Review the structural construction and positional assignments. Check for missing elements or mismatches in reasoning. (e.g., "Claimed 'para hydroxyl' but SMILES places it meta")",
}}
```

C Task description

In this section, we describe the 23 tasks of practical molecular benchmark (Gao et al., 2022). For more details about the task and oracle, refer to Therapeutics Data Commons (Huang et al., 2021, TDC) document: `https://tdc.readthedocs.io/en/main/_modules/tdc/chem_utils/oracle/oracle.html`.

1. albuterol_similarity

Design a molecule similar to albuterol while preserving key functional groups.

2. amlodipine_mpo

Generate molecules similar to amlodipine with good drug-like properties (e.g., 3-ring topology).

3. celecoxib_rediscovery

Recreate the anti-inflammatory drug celecoxib.

4. deco_hop

Modify the decorations of a molecule while preserving a fixed scaffold. Avoid forbidden substructures and stay below similarity cap.

5. drd2

Generate molecules predicted to strongly bind to the dopamine D2 receptor using a predictive model.

6. fexofenadine_mpo

Create molecules structurally similar to fexofenadine with TPSA $\approx$ 90 and logP $\approx$ 4.

7. gsk3b

Design molecules predicted to have high binding affinity for the GSK3β protein.

8. isomers_c7h8n2o2

Generate any molecule that is an exact isomer of $C_7H_8N_2O_2$. Must match the molecular formula exactly.

9. isomers_c9h10n2o2pf2cl

Generate an exact isomer of $C_9H_{10}N_2O_2PF_2Cl$.

10. jnk3

Design molecules with high predicted inhibitory activity against the JNK3 protein.

11. median1

Find a molecule similar to both camphor and menthol.

12. median2

Design a molecule similar to both tadalafil and sildenafil.

13. mestranol_similarity

Generate molecules similar to the hormone mestranol, preserving the core scaffold.

14. osimertinib_mpo

Create osimertinib-like molecules with low logP ($\approx$1) and TPSA $\approx$ 100.

15. perindopril_mpo

Design perindopril-like molecules.

16. qed

Maximize a quantitative estimate of drug-likeness (QED) score.

17. ranolazine_mpo

Create ranolazine-like molecules with TPSA $\approx$ 95 and logP $\approx$ 7.

18. scaffold_hop

Replace the molecular scaffold while keeping key functional groups unchanged.

19. sitagliptin_mpo

Design sitagliptin-like molecules matching formula $C_{16}H_{15}F_6N_5O$.

20. thiothixene_rediscovery

Reproduce the structure of thiothixene.

21. troglitazone_rediscovery

Reconstruct the diabetes drug troglitazone.

22. valsartan_smarts

Generate molecules containing the substructure SMARTS with logP $\approx$ 2.0 and TPSA $\approx$ 95.

23. zaleplon_mpo

Design zaleplon-like molecules with formula $C_{19}H_{17}N_3O_2$.

D PMO-1K experiment result

We provide the full PMO-1K experiment result in Table 10.

E ZINC 250K statistics

We provide the data statistics of ZINC250K (Sterling and Irwin, 2015) that we used in our setting at Table 11.

F Usage of AI assistants

We used AI writing assistants (e.g., ChatGPT) to improve the clarity, grammar, and style of the manuscript during the writing process. These tools were employed strictly for language refinement and did not contribute to the development of ideas, methods, or analysis. All scientific contributions and experimental results are the original work of the authors.

G Scientific Artifacts

The License for artifacts. We used dataset and tools accordingly with their respective licenses. In detail, We use open-source ZINC250K dataset (Sterling and Irwin, 2015) and the publicly available RDKIt tools (Landrum, 2013). We provide our source code at `https://anonymous.4open.science/r/mt_mol-0448` for reproducibility with an appropriate open-source license.

Artifact use consistency with intended use. We used dataset and tools in line of their intended use. Specifically, ZINC250K (Sterling and Irwin, 2015) incorporates molecule with property scores for molecular optimization task which aligns with goal of our study. Also, RDKit tools are used to analyze the chemical properties of the given molecule which is used in our study.

Task	**GP BO**	**REINVENT**	**LICO-*L***	**Genetic GFN**	**Graph GA**	**Aug. Mem.**	**MOLLEO-*B***	**MOLLEO-*D****	**Ours-*D****
albuterol_similarity	0.636 ± 0.106	0.496 ± 0.020	0.656 ± 0.125	0.664 ± 0.054	0.583 ± 0.065	0.557 ± 0.048	$\underline{0.886 \pm 0.023}$	0.883 ± 0.001	$\mathbf{0.998 \pm 0.000}$
amlodipine_mpo	0.519 ± 0.014	0.472 ± 0.008	0.541 ± 0.026	0.534 ± 0.019	0.501 ± 0.016	0.489 ± 0.009	$\underline{0.637 \pm 0.023}$	0.540 ± 0.072	$\mathbf{0.647 \pm 0.010}$
celecoxib_rediscovery	0.411 ± 0.046	0.370 ± 0.029	0.447 ± 0.073	0.447 ± 0.028	0.424 ± 0.049	0.385 ± 0.027	0.402 ± 0.003	$\underline{0.512 \pm 0.119}$	$\mathbf{0.867 \pm 0.007}$
deco_hop	0.593 ± 0.018	0.572 ± 0.006	0.596 ± 0.010	$\underline{0.604 \pm 0.017}$	0.581 ± 0.006	0.579 ± 0.010	0.588 ± 0.007	0.574 ± 0.001	$\mathbf{0.842 \pm 0.077}$
drd2	0.857 ± 0.080	0.775 ± 0.086	$\underline{0.859 \pm 0.066}$	0.809 ± 0.045	0.833 ± 0.065	0.795 ± 0.024	$\mathbf{0.910 \pm 0.017}$	0.812 ± 0.027	0.756 ± 0410
fexofenadine_mpo	$\underline{0.707 \pm 0.021}$	0.650 ± 0.007	0.700 ± 0.023	0.682 ± 0.021	0.666 ± 0.009	0.679 ± 0.021	0.674 ± 0.002	0.680 ± 0.007	$\mathbf{0.883 \pm 0.02}$
gsk3b	0.611 ± 0.059	0.589 ± 0.063	$\underline{0.617 \pm 0.063}$	$\mathbf{0.637 \pm 0.018}$	0.523 ± 0.047	0.539 ± 0.097	0.397 ± 0.013	0.496 ± 0.073	0.308 ± 0.009
isomers_c7h8n2o2	0.545 ± 0.158	0.725 ± 0.064	0.779 ± 0.099	0.738 ± 0.039	0.735 ± 0.112	0.661 ± 0.039	0.737 ± 0.043	$\underline{0.850 \pm 0.009}$	$\mathbf{0.986 \pm 0.015}$
isomers_c9h10n2o2pf2cl	0.599 ± 0.059	0.630 ± 0.032	0.672 ± 0.075	0.656 ± 0.075	0.630 ± 0.086	0.596 ± 0.066	0.635 ± 0.017	$\underline{0.832 \pm 0.007}$	$\mathbf{0.914 \pm 0.031}$
jnk3	$\underline{0.346 \pm 0.067}$	0.315 ± 0.042	0.336 ± 0.051	$\mathbf{0.409 \pm 0.165}$	0.301 ± 0.071	0.294 ± 0.110	0.186 ± 0.076	0.342 ± 0.044	0.125 ± 0.020
median1	0.213 ± 0.020	0.205 ± 0.014	0.217 ± 0.019	0.219 ± 0.008	0.208 ± 0.015	0.219 ± 0.014	$\underline{0.236 \pm 0.021}$	0.193 ± 0.005	$\mathbf{0.321 \pm 0.029}$
median2	0.203 ± 0.009	0.188 ± 0.010	0.193 ± 0.009	$\underline{0.204 \pm 0.011}$	0.181 ± 0.009	0.184 ± 0.010	0.191 ± 0.009	0.197 ± 0.023	$\mathbf{0.322 \pm 0.024}$
mestranol_similarity	0.427 ± 0.025	0.379 ± 0.026	0.423 ± 0.016	0.414 ± 0.022	0.362 ± 0.017	0.393 ± 0.021	0.399 ± 0.020	$\underline{0.630 \pm 0.171}$	$\mathbf{0.996 \pm 0.001}$
osimertinib_mpo	0.766 ± 0.006	0.737 ± 0.007	0.759 ± 0.008	0.763 ± 0.008	0.751 ± 0.005	0.761 ± 0.006	$\underline{0.779 \pm 0.006}$	0.753 ± 0.018	$\mathbf{0.796 \pm 0.005}$
perindopril_mpo	0.458 ± 0.019	0.404 ± 0.008	0.473 ± 0.009	0.462 ± 0.033	0.435 ± 0.016	0.422 ± 0.013	$\mathbf{0.655 \pm 0.054}$	0.422 ± 0.006	$\underline{0.542 \pm 0.027}$
qed	0.912 ± 0.010	0.921 ± 0.002	$\underline{0.925 \pm 0.005}$	$\mathbf{0.928 \pm 0.002}$	0.914 ± 0.007	0.923 ± 0.002	0.919 ± 0.006	$\mathbf{0.928 \pm 0.006}$	0.903 ± 0.003
ranolazine_mpo	$\mathbf{0.701 \pm 0.023}$	0.574 ± 0.044	$\underline{0.687 \pm 0.029}$	0.623 ± 0.022	0.620 ± 0.014	0.614 ± 0.033	0.640 ± 0.000	0.516 ± 0.024	0.233 ± 0.018
scaffold_hop	0.478 ± 0.009	0.447 ± 0.010	0.480 ± 0.008	$\underline{0.485 \pm 0.015}$	0.461 ± 0.008	0.460 ± 0.010	0.473 ± 0.000	0.464 ± 0.002	$\mathbf{0.646 \pm 0.055}$
sitagliptin_mpo	0.232 ± 0.083	0.261 ± 0.026	$\underline{0.315 \pm 0.097}$	0.227 ± 0.041	0.229 ± 0.053	0.245 ± 0.030	0.193 ± 0.073	$\mathbf{0.328 \pm 0.091}$	0.067 ± 0.006
thiothixene_rediscovery	0.351 ± 0.033	0.311 ± 0.021	0.343 ± 0.035	0.377 ± 0.015	0.322 ± 0.023	0.336 ± 0.073	0.416 ± 0.075	$\underline{0.478 \pm 0.028}$	$\mathbf{0.719 \pm 0.001}$
troglitazone_rediscovery	0.313 ± 0.018	0.246 ± 0.009	0.292 ± 0.028	0.277 ± 0.015	0.267 ± 0.015	0.262 ± 0.012	0.302 ± 0.022	$\underline{0.387 \pm 0.013}$	$\mathbf{0.841 \pm 0.042}$
valsartan_smarts	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
zaleplon_mpo	0.392 ± 0.034	0.406 ± 0.017	0.404 ± 0.022	0.400 ± 0.014	0.374 ± 0.024	$\underline{0.415 \pm 0.013}$	0.392 ± 0.003	0.409 ± 0.005	$\mathbf{0.625 \pm 0.046}$
Sum of scores (↑)	11.27	10.68	11.71	11.56	10.90	10.81	11.65	$\underline{12.23}$	**15.42**

Table 10: **Detailed results of PMO-1K benchmark.** Tasks are assessed using AUC top-10 with mean ± standard deviation. Results with (*) are evaluated from 3 independent runs while the others are assessed from 5 independent runs. We mark the best result in **bold** and the second-best are underlined for each task.

Oracle	**Min**	**Max**	**Mean**	**Std**
albuterol_similarity	0.053	0.667	0.251	0.062
amlodipine_mpo	0.000	0.686	0.214	0.144
celecoxib_rediscovery	0.000	0.447	0.142	0.060
deco_hop	0.291	0.878	0.768	0.048
drd2	0.000	0.987	0.009	0.038
fexofenadine_mpo	0.000	0.756	0.232	0.206
gsk3b	0.000	0.990	0.030	0.045
isomers_c7h8n2o2	0.000	1.000	0.004	0.037
isomers_c9h10n2o2pf2cl	0.000	0.869	0.018	0.071
jnk3	0.000	0.680	0.016	0.026
median1	0.000	0.324	0.066	0.037
median2	0.000	0.291	0.108	0.027
mestranol_similarity	0.004	0.886	0.170	0.059
osimertinib_mpo	0.000	0.829	0.179	0.209
perindopril_mpo	0.000	0.560	0.176	0.113
qed	0.117	0.948	0.732	0.139
ranolazine_mpo	0.000	0.586	0.059	0.069
scaffold_hop	0.176	0.526	0.373	0.026
sitagliptin_mpo	0.000	0.479	0.012	0.035
thiothixene_rediscovery	0.000	0.408	0.162	0.047
troglitazon_rediscovery	0.000	0.391	0.135	0.035
valsartan_smarts	0.000	0.320	0.000	0.001
zaleplon_mpo	0.000	0.545	0.072	0.100

Table 11: Data statistics of ZINC 250k that we retrieved for each oracle.