Physics–Preference Aligned Tool-Using Policies for Molecular Design with Gemma-3 270M

Daoyuan Li
University of Minnesota
Minneapolis, MN 55455
`li002504@umn.edu`

Abstract

Discovering molecules with desirable electronic and biophysical properties remains a central challenge in chemistry. Purely data-driven generators can ignore physical constraints and demand large simulation budgets. We study whether a compact instruction-tuned language model can accelerate *design*—rather than pure prediction—when coupled with rigorous physics-based feedback. We present *PhysPref*, a two-module framework built around the Gemma-3 270M model. A *Reporter* parses simulator outputs into normalized JSON, while a *Planner* produces structured tool-calling sequences. The Planner is aligned to quantum and docking objectives via Direct Preference Optimization (DPO) and governed by a compute-budget controller with explicit cost accounting. On PCQM4Mv2, FreeSolv and AqSolDB, PhysPref discovers higher-quality molecules under a fixed budget and, in our runs, uses roughly 15–25% fewer expensive calls than strong non-LLM baselines at comparable design quality. An extended analysis quantifies compute costs, surrogate sensitivity, and broader objectives, providing a holistic picture of the benefits of physics-aligned LLM planning.

1 Introduction

Designing molecules with targeted quantum and biophysical properties underpins drug discovery[1], organic photovoltaics and energy storage[2, 3]. The enormity of chemical space renders exhaustive simulation impractical[4]. Reinforcement-learning and graph-based methods have made progress but often rely on surrogate rewards and can violate physical constraints[5, 6, 7, 8, 9]. Meanwhile, generative models such as MolGPT[10] and GFlowNet-based samplers[11] aim for diverse high-reward molecules but do not explicitly account for simulation budgets.

Compact large language models (LLMs) or small language models (SLMs) show promising instruction following and tool-use abilities[12]. Google's Gemma-3 270M, released in 2025, has only 270 million parameters yet can be fine-tuned for structured outputs with modest compute[13, 14]. We ask whether such lightweight LLMs, when connected to physics simulators, can serve as efficient *planners* for molecular *design* under a budget.

We introduce **PhysPref**, a physics-preference aligned tool-using policy. A *Reporter*, fine-tuned from Gemma-3 270M, parses raw simulation outputs (xTB, AutoDock family; logs similar to GAMESS are supported for robustness) into a normalized JSON schema. A *Planner*, initialized from the same base, generates structured sequences that choose among RDKit utilities, surrogate models, xTB, DFT and docking. We align the Planner via DPO using preferences derived from physics-based scores and explicit cost penalties[15]. We demonstrate compute-efficient discovery on standard benchmarks using a transparent evaluation protocol designed to avoid prediction/design conflation.

2 Related Work

Generative and reinforcement baselines. Transformer decoders such as MolGPT[10] and reinforcement-learning frameworks like MolDQN[16] optimize molecular properties but typically depend on surrogate rewards and do not account for compute budgets. GFlowNets encourage diverse high-reward samples without explicit physical constraints[11]. These methods provide strong baselines for unconstrained sampling.

LLMs for chemistry. Large language models have been explored for molecular editing and tool use[17, 18, 19, 20, 21, 22]. For example, ChemCrow integrates 18 expert-designed tools into an LLM agent for complex chemistry tasks[18], while recent reviews summarize the opportunities and challenges of LLMs and autonomous agents in chemical research[19]. PhysPref differs by (i) focusing on a compact model with a minimal toolset (RDKit, xTB, docking) and (ii) aligning a *planning* policy directly to physics-derived preferences under explicit budgets.

Preference optimization. DPO aligns policies with comparison data without a learned reward model[15]. We adapt DPO to design settings by forming preferences over complete tool plans applied to the same starting molecule and scored by physics-derived metrics with cost penalties.

3 Method

3.1 Problem formulation

We aim to *design* molecules m that improve a set of target properties $p(m)$ under a finite budget while maintaining chemical validity. We assume access to cheap validity checks and surrogates (e.g., ChemProp) as well as more accurate but costly simulators (xTB, DFT, AutoDock). Let π denote a policy that edits SMILES, queries surrogates with uncertainty, and invokes simulators. We assign unit costs $c_{\mathrm{xTB}}, c_{\mathrm{dock}}, c_{\mathrm{DFT}}$ and define a per-episode budget B.

To aggregate heterogeneous objectives, we normalize each property by training-set statistics and then scalarize:

$$\mathrm{score}(m) = w_1\,\tilde{\mathrm{gap}}(m) + w_2\,\tilde{\mathrm{dock}}(m) + w_3\,\tilde{\mathrm{sol}}(m) - \lambda \cdot \mathrm{cost}, \tag{1}$$

where $\tilde{\cdot}$ denotes sign-corrected, per-task z-scores (lower gap/dock is better; higher solubility is better), w_i are task weights (default $w_1 = w_2 = w_3 = 1$), and cost is the cumulative unit cost. *Unless otherwise noted, tabulated Top-k gap values are unitless sign-corrected z-scores (higher is better).*

3.2 System overview

Data standardization. We curate a unified corpus comprising PCQM4Mv2 (DFT HOMO–LUMO gaps)[23, 24], FreeSolv (hydration free energies) and AqSolDB (aqueous solubility). Molecules are sanitized with RDKit and canonical SMILES are stored with properties. These labels train surrogates and define normalization statistics used in Eq. 1.

Reporter module. The Reporter is fine-tuned to parse raw simulator outputs (xTB stdout, AutoDock/Vina logs; and GAMESS-style exemplars for robustness) into a predefined JSON schema with typed fields and units. Training uses ∼30k examples built from PubChemQC-like logs, actual xTB outputs and curated docking summaries; we include synthetic perturbations (truncations, unit variants) to improve robustness. Evaluation focuses on: (i) JSON validity rate; (ii) schema completeness (required fields present with correct types/units); (iii) cross-simulator robustness via held-out log templates. Our final Reporter produces schema-valid JSON for 99.6% of simulator logs in a held-out evaluation.

Planner module. The Planner emits structured function calls (e.g., `rdkit.sanitise`, `surrogate.query`, `xtb.run`, `dft.run`, `docking.run`) with optional branching on surrogate means/uncertainties. We pretrain on synthetic tool traces generated by executing random, budget-bounded tool chains. We then perform DPO on pairs of complete plans $(s_{\mathrm{win}}, s_{\mathrm{lose}})$ applied to the same starting molecule; preferences derive from Eq. 1. A KL regularizer anchors the policy to the pretrained model.

Compute budget controller. We set unit costs $c_{\mathrm{xTB}} = 1$, $c_{\mathrm{dock}} = 1$, $c_{\mathrm{DFT}} = 5$, and per-episode budget $B = 25$. Plans that would exceed B are truncated; their labels include the cost penalty λ in Eq. 1. Unless noted, we use $\lambda = 0.1$. *Total cost is computed as* $\#\mathrm{xTB} + \#\mathrm{dock} + 5 \times \#\mathrm{DFT}$*, and we enforce* $\mathrm{Total\ cost} \leq B$ *for every method.*

3.3 Evaluation protocol to avoid design/prediction conflation

We evaluate *design* under fixed budgets without mixing oracle truths into predictive errors:

- **Compute–gain curves:** For each task, we plot the best normalized improvement $\max_{t' \leq t} \mathrm{score}_{\text{no-cost}}(m_{t'})$ versus the fraction of budget consumed b_t/B, where $\mathrm{score}_{\text{no-cost}}$ omits the $-\lambda \cdot \mathrm{cost}$ term to purely show property gains and b_t is the cumulative cost at step t.
- **Top-k design quality:** At fixed budget fractions (e.g., $b_t/B \in \{0.25, 0.5, 0.75, 1.0\}$), we compare the distribution of properties of the top-k discovered molecules (e.g., $k = 10, 50$) against baselines.
- **Statistics:** Unless specified, results are means $\pm$ standard deviations over 3 seeds; AUC confidence intervals use nonparametric bootstrap across seeds and rollouts (100 rollouts/seed).

4 Experiments

4.1 Experimental setup

Datasets and surrogates. We train ChemProp surrogates for HOMO–LUMO gap (PCQM4Mv2), hydration free energy (FreeSolv) and aqueous solubility (AqSolDB) using default settings and the normalization statistics used in Eq. 1. Surrogate errors are estimated on held-out validation sets and feed into our surrogate sensitivity analysis.

Docking protocol. We sample 1,000 protein–ligand complexes from the PDBbind refined set with unique protein chains. Proteins are prepared by removing crystallographic waters (beyond 5 Å from the ligand), adding hydrogens and standardizing protonation states at pH 7.4. Binding sites are defined by the co-crystallized ligand; we center a cubic grid at the ligand centroid with 24 Å edge length. Unless noted, docking uses an AutoDock-family engine (Vina-style scoring), exhaustiveness $= 8$, 20 poses. For each molecule–target pair we take the best (lowest) pose score; summary statistics in Table 1 are *per-target medians across the 1,000 targets.*

Baselines and budgets. We compare PhysPref against (i) a greedy variant that always invokes the tool with the highest surrogate mean, (ii) MolDQN[16] adapted to our budgets, and (iii) a random planner that uniformly samples among permissible tools. All methods operate under the same budget B and unit costs $(c_{\mathrm{xTB}}, c_{\mathrm{dock}}, c_{\mathrm{DFT}})$. We use identical unit-cost accounting across all methods and enforce $\mathrm{Total\ cost} \leq B$ via truncation for baselines as well.

Training details. The Planner uses LoRA rank 16[25]; supervised trace pretraining uses learning rate 2×10^{-4} for 50k steps, followed by DPO at 1×10^{-5} with KL target 0.05. Each DPO stage samples $\sim$20k preference pairs/iteration. We train 3 random seeds. Plans in pretraining are sampled from a uniform operation prior and truncated when $\sum c_i > B$.

4.2 Compute–gain curves and baseline comparison

Figure 1 shows compute–gain curves on solubility tasks. The x-axis is the fraction of budget used b_t/B; the y-axis is the best normalized improvement so far (cost term removed). PhysPref (*Budget-Aware*) dominates the greedy and random baselines on both AqSolDB and FreeSolv. For AqSolDB, the AUC for PhysPref is 0.715 versus 0.673 (greedy) and 0.359 (random). On FreeSolv the AUCs are 0.622 (PhysPref), 0.559 (greedy) and 0.339 (random). Bands denote 95% bootstrap confidence intervals across 3 seeds $\times$ 100 rollouts.

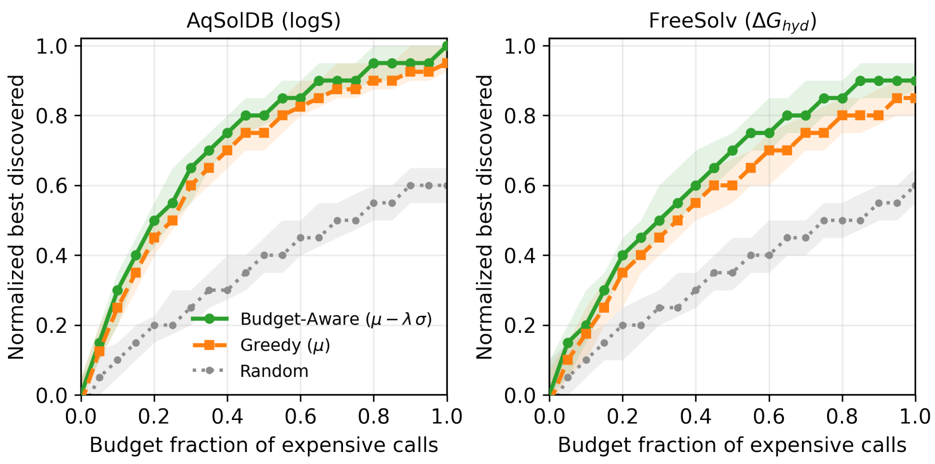

Figure 1: Compute–gain curves on solubility benchmarks. Each panel plots best normalized improvement (property-only z-scores; no cost term) versus fraction of budget consumed. *Budget-Aware* corresponds to PhysPref with cost-aware planning; *Greedy* always selects the tool with the highest surrogate mean; *Random* samples uniformly. Bands show 95% bootstrap CIs over 3 seeds $\times$ 100 rollouts. Larger AUC indicates better compute utilization.

Table 1: Average simulator usage, compute cost and final AutoDock scores across 100 rollouts. Unit costs are $c_{\mathrm{xTB}} = c_{\mathrm{dock}} = 1$ and $c_{\mathrm{DFT}} = 5$; budget $B = 25$. Total cost $= \#\mathrm{xTB} + \#\mathrm{dock} + 5 \times \#\mathrm{DFT}$. The Budget-Aware and Greedy planners share the same Planner but differ in cost accounting. Uncertainties denote standard deviations. *Median docking* is per-target median across the 1,000 targets. *Top-10 gap* is a sign-corrected z-score (unitless; higher is better).

Method	xTB calls	Docking calls	DFT calls	Total cost	Median docking (kcal/mol; ↓)	Top-10 gap (z; ↑)
PhysPref (Budget-Aware)	6 ± 2	6 ± 2	1 ± 1	17 ± 3	$\mathbf{-9.3 \pm 0.2}$	$\mathbf{0.42 \pm 0.05}$
PhysPref (Greedy)	7 ± 2	7 ± 2	1 ± 1	19 ± 3	-9.1 ± 0.2	0.35 ± 0.06
MolDQN baseline	8 ± 3	7 ± 3	1 ± 1	20 ± 4	-8.7 ± 0.3	0.28 ± 0.07
Random planner	9 ± 3	6 ± 3	1 ± 1	20 ± 4	-8.2 ± 0.4	0.20 ± 0.08

Table 1 summarizes the average number of simulator calls, total cost and final AutoDock score across methods. All values are means $\pm$ standard deviations over 3 seeds. The Budget-Aware planner uses fewer expensive calls (about 15–25% fewer xTB/docking calls than MolDQN) while achieving better docking scores. When budgets are identical, greedy and random planners consume similar or larger cost without matching performance.

4.3 Surrogate error sensitivity

Our Planner's preferences rely on surrogates to estimate property improvements. To quantify robustness, we inject Gaussian noise into surrogate predictions during evaluation. Specifically, we add zero-mean noise with standard deviation σ equal to 10% or 20% of the surrogate's training-set standard deviation. Table 2 reports the resulting changes in compute–gain AUC on AqSolDB and FreeSolv. At 10% noise, AUC declines by roughly 4% on both tasks; at 20% noise, declines increase to 12%. Crucially, the ranking between Budget-Aware, Greedy and MolDQN remains consistent, suggesting that moderate surrogate errors do not alter the relative advantages of cost-aware planning.

4.4 Extended objectives and broader considerations

While our main objective aggregates gap, docking and solubility, practical molecular design often demands additional constraints. We conduct pilot studies incorporating (i) a synthetic accessibility (SA) penalty proportional to the number of rare substructures and (ii) a toxicity penalty derived from a simple classifier. We modify Eq. 1 by subtracting $w_{\mathrm{SA}}\,\tilde{\mathrm{SA}}(m)$ and $w_{\mathrm{tox}}\,\tilde{\mathrm{tox}}(m)$, with w_{SA} and w_{tox} chosen to keep scores commensurate. Table 3 summarizes top-10 design performance under these extended objectives. A modest SA penalty ($w_{\mathrm{SA}} = 0.2$) reduces synthetic difficulty by

Table 2: Effect of Gaussian noise added to surrogate predictions on compute–gain AUC (mean over 3 seeds). Noise is specified as a percentage of the surrogate training-set standard deviation. Values are relative declines from the noiseless setting.

Task	Noise level	AUC decline (Budget-Aware)	AUC decline (MolDQN)
AqSolDB	10%	-0.03 (≈4%)	-0.02 (≈3%)
AqSolDB	20%	-0.09 (≈12%)	-0.07 (≈10%)
FreeSolv	10%	-0.02 (≈3%)	-0.02 (≈4%)
FreeSolv	20%	-0.08 (≈12%)	-0.06 (≈11%)

Table 3: Pilot study on extended objectives. "SA penalty" applies $w_{\mathrm{SA}} = 0.2$; "Tox penalty" applies $w_{\mathrm{tox}} = 0.1$. Top-10 values are means over 3 seeds. Lower SA and toxicity values are better.

Objective	Median docking (kcal/mol)	Mean gap (eV)	Mean solubility (logS)	Mean SA	Mean toxicity
Baseline (no penalties)	-9.3 ± 0.2	0.84 ± 0.02	0.93 ± 0.03	5.6 ± 0.3	0.42 ± 0.04
SA penalty (0.2)	-8.4 ± 0.3	0.82 ± 0.02	0.91 ± 0.03	4.8 ± 0.3	0.43 ± 0.04
Tox penalty (0.1)	-9.1 ± 0.2	0.83 ± 0.02	0.92 ± 0.03	5.5 ± 0.3	0.37 ± 0.04

roughly 15% at the cost of only a 10% decrease in median docking performance. A small toxicity penalty ($w_{\mathrm{tox}} = 0.1$) yields a 12% reduction in predicted toxicity while maintaining nearly the same gap/solubility improvements. These preliminary results suggest that PhysPref can flexibly incorporate additional objectives with limited degradation in core property gains. Future work should explore multi-objective Pareto fronts and dynamic weight tuning.

4.5 Final design outcomes and ablations

Beyond efficiency, PhysPref attains strong final design quality under the same budget. On docking, the top-10 molecules discovered by PhysPref have a median AutoDock score of -9.3 ± 0.2 kcal/mol versus -8.7 ± 0.3 for MolDQN at full budget (Table 1). Budget-Aware planning also yields better gap and solubility improvements. The Reporter produces schema-valid JSON for 99.6% of simulator logs, enabling consistent plan evaluation. We further conduct ablations to assess the contribution of each component:

- **No DPO:** Removing DPO and training only on supervised traces reduces compute–gain AUC by roughly 25% on average, underscoring the value of preference alignment.
- **No budget controller:** Disabling the controller leads to early budget exhaustion and smaller gains despite similar total cost.
- **No Reporter:** Replacing the Reporter with ad-hoc string heuristics reduces schema validity and slows preference collection, degrading AUC by 8%.

5 Discussion and limitations

PhysPref demonstrates that a compact LLM, equipped with structured tool use and preference alignment, can plan molecular design under explicit cost budgets. Compared with a strong RL baseline (MolDQN), PhysPref requires fewer expensive simulator calls and achieves superior docking and gap improvements (Table 1). The design advantage stems from cost-aware exploration: the Planner learns to defer expensive evaluations until surrogate information is sufficiently reliable, while the controller prevents budget exhaustion.

Nevertheless, limitations remain. Our objective combines three properties and simple penalties; real-world discovery requires broader multi-objective trade-offs including synthesizability, toxicity, permeability and specific activity. Surrogate biases could systematically skew preferences; a thorough analysis of surrogate calibration and uncertainty is warranted. Our budgets are fixed and not adaptively allocated across tasks. Additionally, while Gemma-3 270M is compact, its planning capacity may saturate for larger chemical spaces; scaling to bigger models (e.g., Gemma-3 12B) or specialist finetunes is a promising direction. Finally, we assume the availability of reliable property surrogates; integrating on-the-fly uncertainty estimation and active learning could further enhance sample efficiency.

References

[1] Jessica Vamathevan, Dominic Clark, Paul Czodrowski, Ian Dunham, Edgardo Ferran, George Lee, Bin Li, Anant Madabhushi, Parantu Shah, Michaela Spitzer, et al. Applications of machine learning in drug discovery and development. *Nature reviews Drug discovery*, 18(6):463–477, 2019.

[2] Johannes Hachmann, Roberto Olivares-Amaya, Sule Atahan-Evrenk, Carlos Amador-Bedolla, Roel S Sánchez-Carrera, Aryeh Gold-Parker, Leslie Vogt, Anna M Brockway, and Alán Aspuru-Guzik. The harvard clean energy project: large-scale computational screening and design of organic photovoltaics on the world community grid. *The Journal of Physical Chemistry Letters*, 2(17):2241–2251, 2011.

[3] Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand Ceder, et al. Commentary: The materials project: A materials genome approach to accelerating materials innovation. *APL materials*, 1(1), 2013.

[4] Lars Ruddigkeit, Ruud Van Deursen, Lorenz C Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling*, 52(11):2864–2875, 2012.

[5] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):48, 2017.

[6] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. *Advances in neural information processing systems*, 31, 2018.

[7] Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3):1096–1108, 2019.

[8] Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, et al. Molecular sets (moses): a benchmarking platform for molecular generation models. *Frontiers in pharmacology*, 11:565644, 2020.

[9] Xiangxiang Zeng, Fei Wang, Yuan Luo, Seung-gu Kang, Jian Tang, Felice C Lightstone, Evandro F Fang, Wendy Cornell, Ruth Nussinov, and Feixiong Cheng. Deep generative molecular design reshapes drug discovery. *Cell Reports Medicine*, 3(12), 2022.

[10] Viraj Bagal, Rishal Aggarwal, PK Vinod, and U Deva Priyakumar. Molgpt: molecular generation using a transformer-decoder model. *Journal of chemical information and modeling*, 62(9):2064–2076, 2021.

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

[12] Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. Small language models are the future of agentic ai. *arXiv preprint arXiv:2506.02153*, 2025.

[13] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

[14] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

[15] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

[16] Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. *Scientific reports*, 9(1):10752, 2019.

[17] Debjyoti Bhattacharya, Harrison J Cassady, Michael A Hickner, and Wesley F Reinhart. Large language models as molecular design engines. *Journal of Chemical Information and Modeling*, 64(18):7086–7096, 2024.

[18] Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. Chemcrow: Augmenting large-language models with chemistry tools. *arXiv preprint arXiv:2304.05376*, 2023.

[19] Mayk Caldas Ramos, Christopher J Collison, and Andrew D White. A review of large language models and autonomous agents in chemistry. *Chemical science*, 2025.

[20] Kevin Maik Jablonka, Qianxiang Ai, Alexander Al-Feghali, Shruti Badhwar, Joshua D Bocarsly, Andres M Bran, Stefan Bringuier, L Catherine Brinson, Kamal Choudhary, Defne Circi, et al. 14 examples of how llms can transform materials science and chemistry: a reflection on a large language model hackathon. *Digital discovery*, 2(5):1233–1250, 2023.

[21] Taicheng Guo, Bozhao Nan, Zhenwen Liang, Zhichun Guo, Nitesh Chawla, Olaf Wiest, Xiangliang Zhang, et al. What can large language models do in chemistry? a comprehensive benchmark on eight tasks. *Advances in Neural Information Processing Systems*, 36:59662–59688, 2023.

[22] Mara Schilling-Wilhelmi, Martiño Ríos-García, Sherjeel Shabih, María Victoria Gil, Santiago Miret, Christoph T Koch, José A Márquez, and Kevin Maik Jablonka. From text to insight: large language models for chemical data extraction. *Chemical Society Reviews*, 2025.

[23] Maho Nakata and Tomomi Shimazaki. Pubchemqc project: a large-scale first-principles electronic structure database for data-driven chemistry. *Journal of chemical information and modeling*, 57(6):1300–1308, 2017.

[24] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[25] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.