ChemSchematic AI: An End-to-End Agentic Framework for Autogenerating Chemical Process Diagrams

Abstract

A clear and well-documented LaTeX document is presented as an article formatted for publication by ACM in a conference proceedings or journal publication. Based on the "acmart" document class, this article presents and explains many of the common variations, as well as many of the formatting elements an author may use in the preparation of the documentation of their work.

Keywords

Chemical Process Diagrams, Process Simulation, Agentic AI, Small Language Models (SLMs), Graph RAG, Model Pruning, KV Caching, Low-Latency Decoding, Test-Time Optimization

1 Introduction

Process Flow Diagrams (PFDs) and Piping and Instrumentation Diagrams (P&IDs) are essential visualization tools in the chemical process industry. A PFD provides a high-level schematic (or layout) of a industrial large-scale chemical production process, of how raw materials are processed and transformed into intermediate and final products. It depicts major equipment, piping, utilities, and key operating conditions of specific equipment within the process, without delving into details of instruments or controls for plant operation. P&IDs, which build upon PFDs, provide a more detailed blueprint of the instrumentation and control systems, which are essential for monitoring, operational control, safety, and maintenance of the chemical process plant. Recent advancements in generative AI are transforming chemical and materials science [7, 15, 19, 22, 33, 44, 50, 53, 53, 56] by accelerating the autonomous discovery of environmental-friendly, next-generation specialty chemicals and the development of low-cost, high-performance material-based products. At the same time, these advancements are augmenting complex first-principles simulation workflows and tedious lab trial-and-error. However, these approaches require development of new production processes for transition novel discoveries from computer simulations or wet-lab experiments into large-scale industrial production, which presents a significant challenge to bring better products to market faster. Generative AI can aid in the conceptualization and design phase of development of novel chemical production processes by autonomously generating high-quality process engineering schematics, providing high-level piping and equipment layouts, along with detailed control instrumentation, which significantly accelerate the industrial scale up of those deemed practical, sustainable, and efficient by swiftly adapting to evolving market and regulatory requirements. Furthermore, integration with first-principle-based process modeling, simulation, and optimization, with reflection, enables continuous self-improvement that can enhance the accuracy of AI-generated PFDs and P&IDs. In this work, we present a closed-loop self-driving lab as an enterprise, cloud-based SaaS Solution for autogenerating PFDs and PIDs for development of novel chemical process as an end-to-end process schematics modeling platform for automatically design, run, analyze and optimize chemical process with minimal human intervention—and continuously learn and improve, on par with human chemical process design engineers, speeds scale up from simulations to market and aligning with the principles of Industry 4.0, which emphasizes smart manufacturing.

Current methods [2, 13, 18, 39, 49], first of all are not designed for auto-generating process schematics or instrumentation and control layouts for novel industrial-scale chemical production processes, severely restricting their utility in early-stage development. These approaches fail to incorporate essential process context—including raw materials, operating conditions, reaction constraints, and end products—rendering them incapable of justifying unit operations in PFDs or defining robust control logic and critical safety interlocks in PIDs. The lack of contextual fidelity, combined with the inability to modify functional or topological features in augmentation approaches and the oversimplified nature of synthetic training data, leads to potentially unrealistic or unsafe designs that fail to reflect the complexity of industrial control architectures required for safe, stable, and efficient chemical process operations. Furthermore, existing methods do not leverage open-source, first-principles-based simulators for verifying the physical and operational feasibility of generated PFDs and PIDs, limiting their industrial reliability. The critique implies that future work should incorporate physics-aware AI, domain knowledge graphs, or simulator-in-the-loop validation to bridge this gap.

2 Methodology

Figure 1 presents a complete pipeline for developing an efficient, feedback-aligned Small Language Model (SLM) tailored to domain-specific process engineering schematics analysis tasks. The pipeline begins with a general-purpose pre-trained SLM, which is first refined using Direct Preference Optimization (DPO)—a preference-based fine-tuning method that aligns the model's responses with human-annotated preferences by contrasting preferred and dispreferred outputs. While DPO improves alignment with human preference supervision, it does not fully teach task-specific reasoning or structural knowledge. To address this, the model is further fine-tuned using either: (i) supervised instruction tuning, which minimizes cross-entropy over instruction–response pairs to inject task knowledge, or (ii) Group Relative Policy Optimization (GRPO), a

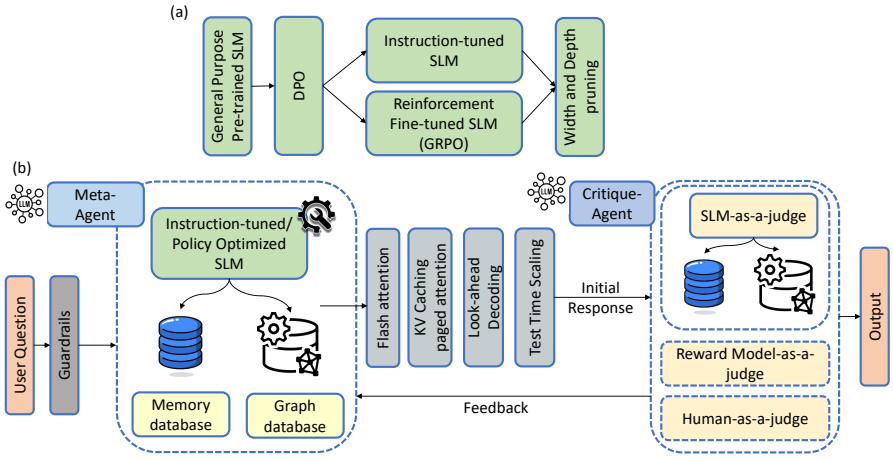


Figure 1: Pipeline for developing and deploying domain-specific Small Language Models for PFD and PID analysis. The framework integrates: (1) model development through DPO alignment and fine-tuning (instruction-tuning or GRPO), with compression via width/depth pruning; (2) Retrieval-Augmented Generation (RAG) architecture with Meta-Agent accessing memory and graph databases for process engineering documentation; (3) performance optimization using Flash attention, KV Caching, Look-ahead Decoding, and test-time scaling strategies; and (4) quality assurance through a Critique-Agent(implementing Nemotron-4-340b-reward, GPT-4o or Anthropic Haiku-As-A-Judge, or human-in-the-loop feedback) that provides feedback for continuous refinement of model responses.

reinforcement-based fine-tuning algorithm that samples multiple candidate completions per prompt, evaluates them using composite reward functions, and updates the policy through group-normalized advantage estimates. To reduce inference costs, the SLM is compressed via width and depth pruning, removing less salient neurons and transformer blocks based on gradient-informed importance scores—yielding a lightweight yet capable model for real-world deployment in chemical process engineering. Figure 1 presents the complete architecture of the proposed Retrieval-Augmented Generation (RAG) framework tailored for automated interpretation, analysis, and synthesis of PFDs and PIDs in chemical process engineering. At its core, a meta-agent orchestrates task decomposition, tool (external database) selection, and retrieval from both: a dense memory database that stores prior conversational history and contextual knowledge, and a graph database constructed from structured industrial chemical production processes compiled by an AI-driven agentic web navigation framework that autonomously retrieves, interprets, and synthesizes data from diverse sources. The meta-agent leverages the fine-tuned SLM as its reasoning engine, and the retrieved context is fed to the SLM for response generation. For inference-time performance, the framework integrates low-latency decoding techniques including FlashAttention for block-wise attention computation, paged key–value caching for memory-efficient reuse of context, and Lookahead Decoding for speculative multi-token generation with fidelity-preserving verification. Additionally, test-time scaling strategies such as self-consistency decoding, confidence-weighted entropy scoring, and self-reflection are employed to enhance reasoning quality and stability. Output refinement is governed by a critique agent, implemented using either a reward model (Nemotron-4-340b-reward), LLM-as-a-judge (GPT-4o or Anthropic Haiku), or human-in-the-loop feedback, which evaluates initial outputs and triggers revision when necessary. Altogether, this multi-stage architecture integrates symbolic graph-based reasoning, dense memory retrieval, high-throughput decoding, and iterative output refinement—making it especially effective for industrial-scale generation and analysis of chemical process documentation.

3 Experiments

3.1 Datasets

In this work, we curated a chemical database of over 1,120 chemicals spanning key sectors such as electronics, energy, and pharmaceuticals, with a focus on those critical to modern manufacturing and utilities. The data were extracted from product catalogs of major manufacturers—including BASF, Dow Chemical, DuPont, Solvay, Mitsubishi Chemical, Bayer, Evonik, SABIC, and LyondellBasell—ensuring high fidelity, reliability, and comprehensive coverage of industrial chemicals. The dataset comprises two components. The first, **ChemAtlas**, is a core collection of 1,020 chemicals. We utilize an AI-driven agentic web navigation framework that autonomously retrieves, interprets, and synthesizes data from diverse sources to compile detailed information on industrial chemicals, including metadata on their industrial production processes as well as their associated PFDs and PIDs. This structured information serves as the foundation for constructing ontological knowledge graphs. We then employed advanced LLMs (GPT-4o and Anthropic Claude Haiku) to generate and cross-validate chemical-specific PFDs and PIDs, leveraging their pre-trained knowledge for consistency checks. The second component,

ChemEval, is a secondary evaluation set of 100 chemicals specifically curated to test the framework's robustness and generalization when auto-generating PFDs and PIDs for novel, unseen chemicals. We also employ a teacher-student transfer learning approach, generating custom synthetic datasets using the ChemAtlas database, with 20,000 QA pairs created by large language models (LLMs) such as GPT-4o and Anthropic Haiku based on predefined templates. The textual outputs are scored, validated, and filtered using Nvidia-Nemotron-4-340B to train smaller language models (SLMs) (e.g., Llama-3.2-1B, SmolLM-135M) for domain-specific tasks related to interpreting, analyzing, and auto-generating PFDs and PIDs. These datasets comprise QA pairs focused on factual knowledge, preference optimization, chemical process synthesis descriptions, logical reasoning, multi-step chain-of-thought reasoning, sensor and instrumentation planning, comparative process analysis, and error detection and correction. These diverse datasets are designed to enhance small-scale models' capabilities in designing piping and equipment layouts, planning sensor and instrumentation layouts, and other related tasks—thereby enabling a deeper understanding of complex relationships within process engineering tasks.

3.2 Experimental Setup

The curated collection consisted of six specialized synthetic subsets, each targeting a distinct model capability. Factual QA dataset, generated via hierarchical topic decomposition, were curated to improve factual recall and question-answering performance with focus on process engineering schematics. The SynDIP dataset comprised validated sequential trajectories linking process blueprints (industrial production methods, including raw materials, reaction steps, and major equipment), PFD descriptions, and PID descriptions. The LogiCore dataset comprised multi-step reasoning QA pairs grounded in PFD and PID schematics to justify process design decisions, validate control logic, and explain flow sequencing in chemical process diagrams. The DPO dataset comprised preference-labeled QA pairs, where each pair included preferred and dispreferred model-generated responses identified using reward model score differentials, enabling alignment tuning via Direct Preference Optimization. The RAIT (Retrieval-Augmented Instruction Tuning) datasets conditioned generation on retrieved content from the SynDIP corpus. Local RAIT used QA pairs grounded in individual text chunks, while Global RAIT used semantically clustered groups of chunks for document-level comprehension. The framework performance was assessed on two distinct datasets generated by teacher models (e.g., o3 and o4-mini from OpenAI, and Anthropic Sonnet), using a self-instruct bootstrapping method. A small seed set of human-written instruction-response examples was used to initiate the generation of high-quality question-answer pairs through iterative synthesis. One of these datasets—the Held-out Evaluation Set—comprised 1,500 QA pairs based on unseen prompts and was used to evaluate generalization across core capabilities, including factual knowledge, reasoning, instruction following, and PFD/PID interpretation. In addition to general capabilities, we assessed the framework's ability to generate process and control schematics for compounds not present in ChemAtlas using the ChemEval dataset. For each compound, the teacher models generated PFD and PID descriptions in the form of QA pairs using the same self-instruct bootstrapping method. These served as reference targets for quantitative evaluation of the framework. We assessed model performance through a combination of quantitative, qualitative, and system-level evaluations. Quantitative analysis measured textual similarity and overlap using standard NLP metrics: BLEU, ROUGE (variants), METEOR, SacreBLEU, BERTScore, and Semantic Similarity (via sentence transformers). Qualitative performance, particularly regarding correctness, coherence, and helpfulness, was evaluated using the Nemotron-4-340B reward model, which provided scores (0-4 scale) across five dimensions (Helpfulness, Correctness, Coherence, Complexity, Verbosity) for responses generated on the held-out test set and the ChemEval generalization benchmark. The efficiency impact of inference optimizations (Paged Attention, Lookahead Decoding, FlashAttention-2) was quantified using system metrics: inference throughput (tokens/sec), generation latency (sec), maximum batch size, and peak GPU memory usage (GB). Finally, to establish practical relevance, we performed engineering verification using DWSIM, an open-source chemical process simulator. Autogenerated PFD/PID textual descriptions were manually translated into executable DWSIM flowsheets. These flowsheets enabled steady-state or dynamic simulations to rigorously validate core engineering principles, including material/energy balances and thermodynamic consistency (verifying PFD aspects). Functional validation of specified control strategies (PID aspects) was conducted indirectly by manually configuring control loops within DWSIM's dynamic simulation environment and assessing their stability and response characteristics (e.g., setpoint tracking, disturbance rejection). For the Graph RAG, our experimental setup involved constructing a heterogeneous knowledge graph. This graph was populated using domain-specific documents (web data related to chemical production processes for chemicals in our ChemAtlas database) gathered via our agentic web navigation framework. Text from these retrieved documents was segmented into chunks using a sliding window approach. We utilized GPT-4o to extract subject-predicate-object triples from these chunks, forming the semantic relation edges within the graph. Structural containment edges linked identified entity nodes back to their source chunk nodes. To ensure entity consistency, we implemented a canonicalization step, merging entity nodes based on high semantic similarity (using text-embedding-3-small embeddings) and high lexical similarity (string edit distance), applying predefined thresholds. For efficient retrieval, the graph was partitioned into hierarchical communities using the Leiden algorithm, optimizing for modularity. During experiments, the retrieval mechanism ranked communities based on the cosine similarity between the input query embedding and pre-computed community summaries

(also generated using GPT-4o). The structured context provided to the reasoning SLM for inference was the subgraph induced by the top-K selected communities. This subgraph explicitly included the relevant entity nodes, the semantic relationship triples connecting them, and the associated text chunk nodes that provided the source textual context. All fine-tuning experiments for the Llama-3.2 1B and SmolLM2-135M models utilized Quantized Low-Rank Adaptation (QLoRA) [**?**], employing frozen 4-bit NF4 quantized base weights alongside trainable low-rank adapters applied to key transformer projection layers. Training consistently used the 8-bit AdamW optimizer with a learning rate of 2×10^{-4} (linear decay, 0.01 weight decay), an effective batch size of 8 (per-device batch size 2, 4 accumulation steps), a maximum sequence length of 4096 tokens via gradient checkpointing, and mixed precision (BF16/FP16) on NVIDIA H100 GPUs. We investigated two primary fine-tuning methodologies. The first was a sequential pipeline: Supervised Fine-Tuning (SFT) on the combined Factual QA, SynDIP, and LogiCore datasets (15 epochs), followed by Direct Preference Optimization (DPO) using the DPO dataset (5 epochs), and concluding with Retrieval-Augmented Instruction Tuning (RAIT) using the Local/Global RAIT datasets (15 epochs). Alternatively, Group Relative Policy Optimization (GRPO) was investigated. We applied GRPO starting from two distinct model states: first, directly from the pretrained base model using the combined Factual QA, SynDIP, and LogiCore datasets; and second, starting from the model state obtained after the aforementioned SFT stage, using the Local/Global RAIT datasets. Both GRPO applications utilized the established QLoRA configuration and optimized a clipped surrogate objective [**?**] based on normalized advantages derived from a composite reward function (weighting ROUGE-L, length ratio, and an LLM score) evaluated over sampled response groups ($G = 4$). KL divergence regularization was applied against the appropriate reference policy (pretrained state for the first setting, SFT state for the second) to maintain stability and retain foundational knowledge. GRPO training proceeded for approximately 10-15 epochs until convergence in both settings. Implementations leveraged PyTorch, Hugging Face libraries (`transformers`, `datasets`, `peft`, `trl`). To comprehensively evaluate a suite of modern optimization techniques designed to make fine-tuned SLMs more efficient and reliable, we configured specific experiments using the Llama-3.2 1B model on NVIDIA H100 GPUs. This evaluation systematically isolates and quantifies the impact of model pruning for compression, Paged Attention for memory efficiency, Lookahead Decoding and FlashAttention-2 for reducing latency, and test-time inference scaling for improving output quality without retraining. The goal was to understand how these complementary strategies contribute individually to improving the model's suitability for complex engineering tasks. To assess the trade-off between model compression and predictive performance, we systematically applied width and depth pruning. The objective was to reduce model size, memory footprint, and potentially inference latency by removing less critical neurons (width) or entire layers (depth). Pruning levels (e.g., 1% to 50%) were tested, using gradient-based importance scores calculated during the concurrent QLoRA fine-tuning process to guide removal To enhance the reliability, factual accuracy, and reasoning robustness of the model's outputs at inference time without modifying model parameters, we employed a test-time scaling mechanism. This multi-stage process (including Chain-of-Thought sampling (N=10), confidence-weighted ranking (lambda=0.5, K=3), self-reflection/revision, and self-consistency voting) aims to produce more dependable answers by exploring, evaluating, refining, and consolidating multiple reasoning paths. Its effectiveness was evaluated by comparing the qualitative reward model scores of its outputs against those from baseline greedy decoding. To improve inference throughput and memory utilization by addressing cache fragmentation, we benchmarked Paged Attention. Its purpose is to enable larger batch sizes and handle longer sequences more effectively than standard contiguous KV caching. We compared its performance using system metrics (max batch size, throughput, memory usage) against the baseline implementation. To reduce end-to-end generation latency during inference, Lookahead Decoding was implemented. This technique aims to accelerate decoding by generating multiple tokens in parallel per model forward pass while guaranteeing the same output sequence as standard greedy decoding. We configured it with specific lookahead parameters (e.g., N=5 iterations, L=10 positions) and measured its impact on latency and throughput compared to the baseline. To accelerate the core attention computation and reduce memory bandwidth usage during both training and inference, we integrated FlashAttention-2. Its advantage lies in optimizing GPU memory access patterns and measured the resulting changes in training speed, peak training memory, and inference speed.

3.3 Results

Figures 2a, 2b, and 2c (for Llama 3.2 1B) alongside Figures 2d, 2e, and 2f (for SmolLM2-135M) present evaluation metrics (scaled from 0 to 1) for n-gram overlap (BLEU, ROUGE, Meteor, SacreBLEU) and semantic similarity (BERTScore, Semantic Similarity) under three fine-tuning regimes: supervised QA (on Factual QA, SynDIP, LogiCore), DPO (on a preference-annotated dataset), and multi-scale RAIT (on Local and Global RAIT datasets). For Llama 3.2 1B, Figure 2a shows supervised QA fine-tuning achieves very high semantic scores alongside moderate-to-low n-gram overlap scores. In contrast, Figures 2b and 2c reveal that both DPO and RAIT fine-tuning lead to significantly lower n-gram scores compared to QA fine-tuning, while semantic scores remain high but slightly below the QA baseline. A different trend is observed for SmolLM2-135M: Figure 2d indicates that supervised QA fine-tuning yields the strongest performance for this model, particularly in n-gram metrics and Similarity, though with a moderate BERTScore. However, Figures 2e and 2f show that DPO and especially RAIT fine-tuning result in decreased performance across most metrics compared to the SmolLM QA baseline, with RAIT fine-tuning producing substantially lower scores overall. Thus, the

impact of DPO and RAIT differs notably between the models, reducing n-gram performance while largely maintaining semantic scores for Llama 3.2 1B, but generally degrading performance across the board for SmolLM2-135M relative to their respective QA fine-tuned versions. We now aim to evaluate the impact of fine-tuning and retrieval-augmented generation (Graph RAG) on the qualitative performance of six language model variants, spanning two model sizes—Llama-1B and SmolLM-135M. Each variant differs based on whether fine-tuning (FT) and Graph RAG are applied. The models are assessed using the Nvidia/Nemotron-4-340B reward model across five key qualitative metrics—helpfulness, correctness, coherence, complexity, and verbosity—as shown in Figures 3a–e. In this study, we compare six model variants, where W/ stands for “with", W/o stands for “without", FT indicates “fine tuning", and Graph RAG refers to “Graph Retrieval-Augmented Generation". The six variants are: (a) Llama-3.2 1B With Fine Tuning With Graph RAG (Llama-3.2 1B W/FT W/Graph RAG), (b) Llama-3.2 1B With Fine Tuning Without Graph RAG (Llama-3.2 1B W/FT W/o Graph RAG), (c) Llama-3.2 1B Without Fine Tuning Without Graph RAG (Llama-3.2 1B W/o FT W/o Graph RAG), (d) Llama-3.2 1B Without Fine Tuning With Graph RAG (Llama-3.2 1B W/o FT W/Graph RAG), (e) SmolLM2-135M With Fine Tuning With Graph RAG (SmolLM2-135M W/FT W/GraphRAG), and (f) SmolLM2-135M With Fine Tuning Without Graph RAG (SmolLM2-135M W/FT W/o GraphRAG). Among these, the larger Llama-3.2 1B models yield consistently stronger performance, benefiting especially from fine-tuning and retrieval augmentation. Variant (a), Llama-3.2 1B W/FT W/Graph RAG, provides the most balanced and high-scoring responses, particularly in correctness and complexity, though it may be more verbose due to incorporating additional retrieved content. Variant (b), Llama-3.2 1B W/FT W/o Graph RAG, remains comparably coherent and helpful but can lag on fact-heavy tasks lacking external retrieval. Even without fine-tuning, variant (d), Llama-3.2 1B W/o FT W/Graph RAG, demonstrates that retrieval augmentation improves correctness for knowledge-intensive prompts, whereas variant (c), Llama-3.2 1B W/o FT W/o Graph RAG, represents the most basic Llama variant. For the smaller SmolLM2-135M models, retrieval also improves correctness—making variant (e), SmolLM2-135M W/FT W/GraphRAG, more effective than its counterpart (f), SmolLM2-135M W/FT W/o GraphRAG—though both underperform the Llama-3.2 1B family in coherence, complexity, and helpfulness. In conclusion, the results demonstrate that fine-tuning and retrieval significantly enhance output quality, particularly for larger models. Among all configurations, Llama W/FT W/Graph RAG (variant A) achieves the most balanced and high-scoring responses in terms of correctness, complexity, and helpfulness, though it also tends to be more verbose. The study highlights the complementary roles of fine-tuning and retrieval in improving the coherence and factual accuracy of responses, while also benchmarking performance trade-offs between large and compact models. To summarize, fine-tuning substantially improves model performance across multi-dimensional metrics (e.g., correctness, coherence, and complexity) by enabling more accurate and structured responses. Graph RAG provides additional gains—particularly for factual tasks—by enhancing correctness through external knowledge retrieval, though its impact varies by model size. For non-fine-tuned models (e.g., Llama-3.2 1B W/o FT), retrieval partially compensates for the lack of fine-tuning, while models lacking both methods perform the worst. This highlights the limitations of intrinsic pretraining knowledge alone, especially in specialized domains like chemical process synthesis where retrieval and task-specific tuning are critical.

4 Conclusion

References

[1] Vincent Abbott and Gioele Zardini. 2024. FlashAttention on a Napkin: A Diagrammatic Approach to Deep Learning IO-Awareness. *arXiv preprint arXiv:2412.03317* (2024).

[2] Achmad Anggaviya Alimin, Dominik P. Goldstein, Lukas Schulze Balhorn, and Artur M. Schweidtmann. 2025. Talking like Piping and Instrumentation Diagrams (P&IDs). *arXiv preprint arXiv:2502.18928* (2025).

[3] Vidhisha Balachandran, Jingya Chen, Lingjiao Chen, Shivam Garg, Neel Joshi, Yash Lara, John Langford, Besmira Nushi, Vibhav Vineet, Yue Wu, et al. 2025. Inference-Time Scaling for Complex Tasks: Where We Stand and What Lies Ahead. *arXiv preprint arXiv:2504.00294* (2025).

[4] Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. 2024. Forest-of-thought: Scaling test-time compute for enhancing LLM reasoning. *arXiv preprint arXiv:2412.09078* (2024).

[5] Jiefeng Chen, Jie Ren, Xinyun Chen, Chengrun Yang, Ruoxi Sun, and Sercan Ö Arık. 2025. SETS: Leveraging Self-Verification and Self-Correction for Improved Test-Time Scaling. *arXiv preprint arXiv:2501.19306* (2025).

[6] Shimao Chen, Zirui Liu, Zhiying Wu, Ce Zheng, Peizhuang Cong, Zihan Jiang, Yuhan Wu, Lei Su, and Tong Yang. 2024. INT-FlashAttention: Enabling Flash Attention for INT8 Quantization. *arXiv preprint arXiv:2409.16997* (2024).

[7] Yuan Chiang, Elvis Hsieh, Chia-Hong Chou, and Janosh Riebesell. 2024. LLAMP: Large language model made powerful for high-fidelity materials knowledge retrieval and distillation. *arXiv preprint arXiv:2401.17244* (2024).

[8] Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691* (2023).

[9] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems* 35 (2022), 16344–16359.

[10] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130* (2024).

[11] Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. 2024. Break the sequential dependency of llm inference using lookahead decoding. *arXiv preprint arXiv:2402.02057* (2024).

[12] Yuan Gao, Zujing Liu, Weizhong Zhang, Bo Du, and Gui-Song Xia. 2024. Bypass Back-propagation: Optimization-based Structural Pruning for Large Language Models via Policy Gradient. *arXiv preprint arXiv:2406.10576* (2024).

[13] Shreesha Gowiakar, Srinivasan Iyengar, Sameer Segal, and Shivkumar Kalyanaraman. 2024. An Agentic Approach to Automatic Creation of P&ID Diagrams from Natural Language Descriptions. *arXiv preprint arXiv:2412.12898* (2024).

[14] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* (2025).

[15] Jeff Guo and Philippe Schwaller. 2024. Saturn: Sample-efficient generative molecular design using memory manipulation. *arXiv preprint arXiv:2405.17066* (2024).

[16] Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, et al. 2024. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309* (2024).

[17] Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *Advances in Neural Information Processing Systems* 37 (2024), 132876–132907.

[18] Edwin Hirretier, Lukas Schulze Balhorn, and Artur M. Schweidtmann. 2022. Towards automatic generation of piping and instrumentation diagrams (P&IDs) with artificial intelligence. *arXiv preprint arXiv:2211.05583* (2022).

[19] Chenglong Kang, Xiaoyi Liu, and Fei Guo. [n. d.]. RetroInText: A Multimodal Large Language Model Enhanced Framework for Retrosynthetic Planning via In-Context Representation Learning. In *The Thirteenth International Conference on Learning Representations*.

[20] Aum Kendapadi, Kerem Zaman, Rakesh R Menon, and Shashank Srivastava. 2024. INTERACT: Enabling Interactive, Question-Driven Learning in Large Language Models. *arXiv preprint arXiv:2412.11388* (2024).

[21] Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. 2024. Shortened LLaMA: Depth Pruning for Large Language Models with Comparison of Retraining Methods. *arXiv preprint arXiv:2402.02834* (2024).

[22] Agustinus Kristiadi, Felix Strieth-Kalthoff, Marta Skreta, Pascal Poupart, Alán Aspuru-Guzik, and Geoff Pleiss. 2024. A sober look at LLMs for material discovery: Are they actually good for Bayesian optimization over molecules? *arXiv preprint arXiv:2402.05015* (2024).

[23] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*. 611–626.

[24] Xinzhe Li. 2025. A Survey on LLM Test-Time Compute via Search: Tasks, LLM Profiling, Search Algorithms, and Relevant Frameworks. *arXiv preprint arXiv:2501.10069* (2025).

[25] Zhihang Lin, Mingbao Lin, Yuan Xie, and Rongrong Ji. 2025. CPPO: Accelerating the Training of Group Relative Policy Optimization-Based Reasoning Models. *arXiv preprint arXiv:2503.22342* (2025).

[26] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437* (2024).

[27] Qin Liu, Wenxuan Zhou, Nan Xu, James Y Huang, Fei Wang, Sheng Zhang, Hoifung Poon, and Muhao Chen. 2025. Metascale: Test-time scaling with evolving meta-thoughts. *arXiv preprint arXiv:2503.13447* (2025).

[28] Yue Liu, Jiaying Wu, Yufei He, Hongcheng Gao, Hongyu Chen, Baolong Bi, Jiaheng Zhang, Zhiqi Huang, and Bryan Hooi. 2025. Efficient Inference for Large Reasoning Models: A Survey. *arXiv preprint arXiv:2503.23077* (2025).

[29] Haiquan Lu, Yefan Zhou, Shiwei Liu, Zhangyang Wang, Michael W Mahoney, and Yaoqing Yang. 2024. Alphapruning: Using heavy-tailed self regularization theory for improved layer-wise pruning of large language models. *Advances in Neural Information Processing Systems* 37 (2024), 9117–9152.

[30] Jonathan Mamou, Oren Pereg, Daniel Korat, Moshe Berchansky, Nadav Timor, Moshe Wasserblat, and Roy Schwartz. 2024. Dynamic speculation lookahead accelerates speculative decoding of large language models. *arXiv preprint arXiv:2405.04304* (2024).

[31] Daniel Medeiros. 2025. DWSIM: Open Source Process Simulator. https://dwsim.fossee.in Accessed April 15, 2025.

[32] OpenAI. 2024. text-embedding-3-small model. https://platform.openai.com/docs/guides/embeddings. Accessed: August 2024.

[33] Elton Pan, Soohyoung Kwon, Sulin Liu, Mingrou Xie, Yifei Duan, Thorben Prein, Killian Sheriff, Yuriy Roman, Manuel Moliner, Rafael Gómez-Bombarelli, et al. 2024. A chemically-guided generative diffusion model for materials synthesis planning. In *AI for Accelerated Materials Design–NeurIPS 2024*.

[34] Ramya Prabhu, Ajay Nayak, Jayashree Mohan, Ramachandran Ramjee, and Ashish Panwar. 2024. vattention: Dynamic memory management for serving llms without pagedattention. *arXiv preprint arXiv:2405.04437* (2024).

[35] Yuxiao Qu, Matthew YR Yang, Amrith Setlur, Lewis Tunstall, Edward Emanuel Beeching, Ruslan Salakhutdinov, and Aviral Kumar. 2025. Optimizing test-time compute via meta reinforcement fine-tuning. *arXiv preprint arXiv:2503.07572* (2025).

[36] Ankit Singh Rawat, Veeranjaneyulu Sadhanala, Afshin Rostamizadeh, Ayan Chakrabarti, Wittawat Jitkrittum, Vladimir Feinberg, Seungyeon Kim, Hrayr Harutyunyan, Nikunj Saunshi, Zachary Nado, et al. 2024. A little help goes a long way: Efficient llm training by leveraging small lms. *arXiv preprint arXiv:2410.18779* (2024).

[37] Isaac Rehg. 2024. KV-Compress: Paged KV-Cache Compression with Variable Compression Rates per Attention Head. *arXiv preprint arXiv:2410.00161* (2024).

[38] Fabrizio Sandri, Elia Cunegatti, and Giovanni Iacca. 2025. 2SSP: A Two-Stage Framework for Structured Pruning of LLMs. *arXiv preprint arXiv:2501.17771* (2025).

[39] Lukas Schulze Balhorn, Edwin Hirretier, Lynn Luderer, and Artur M Schweidtmann. 2023. Data augmentation for machine learning of chemical process flowsheets. *arXiv e-prints* (2023), arXiv–2302.

[40] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. 2024. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *Advances in Neural Information Processing Systems* 37 (2024), 68658–68685.

[41] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300* (2024).

[42] Nishad Singhi, Hritik Bansal, Arian Hosseini, Aditya Grover, Kai-Wei Chang, Marcus Rohrbach, and Anna Rohrbach. 2025. When To Solve, When To Verify: Compute-Optimal Problem Solving and Generative Verification

for LLM Reasoning. *arXiv preprint arXiv:2504.01005* (2025).

[43] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314* (2024).

[44] Henry W Sprueill, Carl Edwards, Khushbu Agarwal, Mariefel V Olarte, Udishnu Sanyal, Conrad Johnston, Hongbin Liu, Heng Ji, and Sutanay Choudhury. 2024. ChemReasoner: Heuristic search over a large language model's knowledge space using quantum-chemical feedback. *arXiv preprint arXiv:2402.10980* (2024).

[45] Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695* (2023).

[46] Wenfang Sun, Xinyuan Song, Pengxiang Li, Lu Yin, Yefeng Zheng, and Shiwei Liu. 2025. The Curse of Depth in Large Language Models. *arXiv preprint arXiv:2502.05795* (2025).

[47] Shengkun Tang, Oliver Sieberling, Eldar Kurtic, Zhiqiang Shen, and Dan Alistarh. 2025. DarwinLM: Evolutionary Structured Pruning of Large Language Models. *arXiv preprint arXiv:2502.07780* (2025).

[48] Yijun Tian, Yikun Han, Xiusi Chen, Wei Wang, and Nitesh V Chawla. 2025. Beyond answers: Transferring reasoning capabilities to smaller llms using multi-teacher knowledge distillation. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*. 251–260.

[49] Gabriel Vogel, Lukas Schulze Balhorn, and Artur M Schweidtmann. 2023. Learning from flowsheets: A generative transformer model for autocompletion of flowsheets. *Computers & Chemical Engineering* 171 (2023), 108162.

[50] Haorui Wang, Marta Skreta, Cher-Tian Ser, Wenhao Gao, Lingkai Kong, Felix Strieth-Kalthoff, Chenru Duan, Yuchen Zhuang, Yue Yu, Yangqiao Zhu, et al. 2024. Efficient evolutionary search over chemical space with large language models. *arXiv preprint arXiv:2406.16976* (2024).

[51] Haihang Wu. 2024. LLM-BIP: Structured Pruning for Large Language Models with Block-Wise Forward Importance Propagation. *arXiv preprint arXiv:2412.06419* (2024).

[52] Junjie Yang, Junhao Song, Xudong Han, Ziqian Bi, Tianyang Wang, Chia Xin Liang, Xinyuan Song, Yichao Zhang, Qian Niu, Benji Peng, et al. 2025. Feature Alignment and Representation Transfer in Knowledge Distillation for Large Language Models. *arXiv preprint arXiv:2504.13825* (2025).

[53] Sherry Yang, Simon Batzner, Ruiqi Gao, Muratahan Aykol, Alexander Gaunt, Brendan C McMorrow, Danilo Jimenez Rezende, Dale Schuurmans, Igor Mordatch, and Ekin Dogus Cubuk. 2024. Generative hierarchical materials search. *Advances in Neural Information Processing Systems* 37 (2024), 38799–38819.

[54] Wenkai Yang, Shuming Ma, Yankai Lin, and Furu Wei. 2025. Towards thinking-optimal scaling of test-time compute for llm reasoning. *arXiv preprint arXiv:2502.18080* (2025).

[55] Zhaojian Yu, Yinghao Wu, Yilun Zhao, Arman Cohan, and Xiao-Ping Zhang. 2025. Z1: Efficient Test-time Scaling with Code. *arXiv preprint arXiv:2504.00810* (2025).

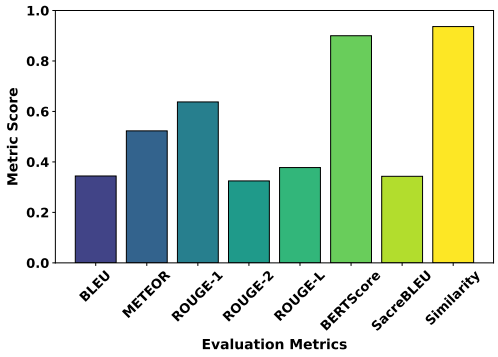
[56] Huan Zhang, Yu Song, Ziyu Hou, Santiago Miret, and Bang Liu. 2024. Honeycomb: A flexible llm-based agent system for materials science. *arXiv preprint arXiv:2409.00155* (2024).

[57] Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Zhihan Guo, Yufei Wang, Irwin King, Xue Liu, and Chen Ma. 2025. What, How, Where, and How Well? A Survey on Test-Time Scaling in Large Language Models. *arXiv preprint arXiv:2503.24235* (2025).

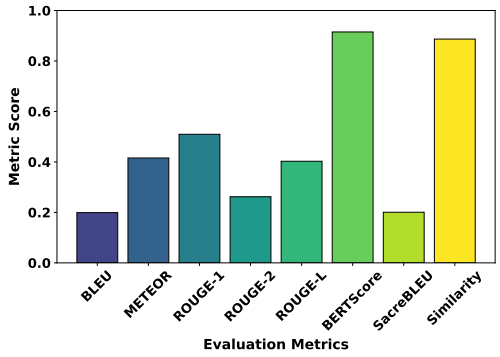
[58] Yao Zhao, Zhitian Xie, Chen Liang, Chenyi Zhuang, and Jinjie Gu. 2024. Lookahead: An inference acceleration framework for large language model with lossless generation accuracy. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 6344–6355.

[59] Ming Zhong, Chenxin An, Weizhu Chen, Jiawei Han, and Pengcheng He. 2023. Seeking Neural Nuggets: Knowledge Transfer in Large Language Models from a Parametric Perspective. *arXiv preprint arXiv:2310.11451* (2023).

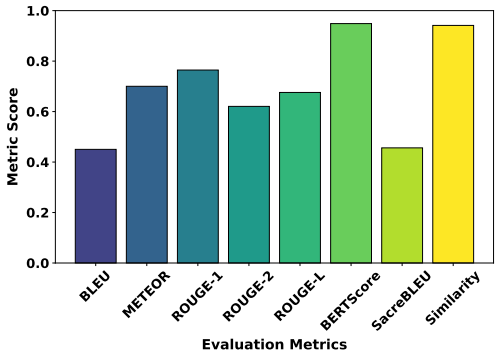
[60] Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2024. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics* 12 (2024), 1556–1577.



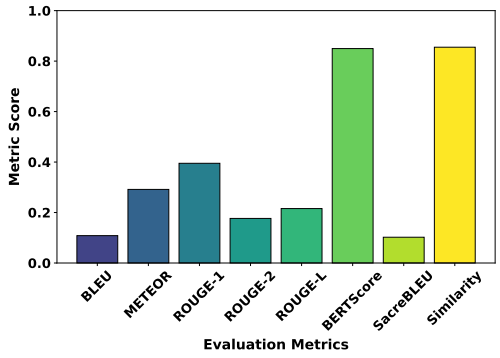
(a) Evaluation on test sets for Llama 3.2 1B, fine-tuned on the training splits of knowledge-intensive QA datasets (Factual QA, SynDIP, LogiCore). Metrics include BLEU, ROUGE, BERTScore, and Semantic Similarity.



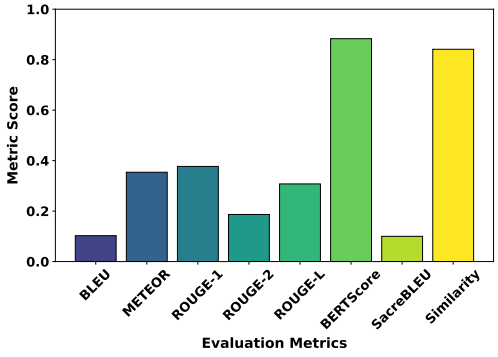
(b) Evaluation on test sets for Llama 3.2 1B, fine-tuned via DPO on preference-annotated training data. Reported metrics include BLEU, ROUGE, Meteor, BERTScore, SacreBLEU, and Semantic Similarity.



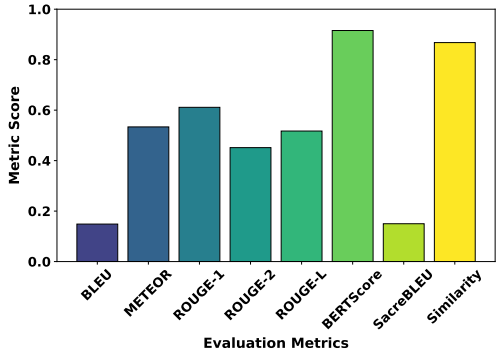
(c) Evaluation on test sets for Llama 3.2 1B, fine-tuned on the training splits of multi-scale RAIT datasets (Local and Global). Metrics include BLEU, ROUGE, BERTScore, and Semantic Similarity.



(d) Evaluation on test sets for SmolLM2-135M, fine-tuned on the training splits of QA datasets (Factual QA, SynDIP, LogiCore). Metrics include BLEU, ROUGE, BERTScore, and Semantic Similarity.

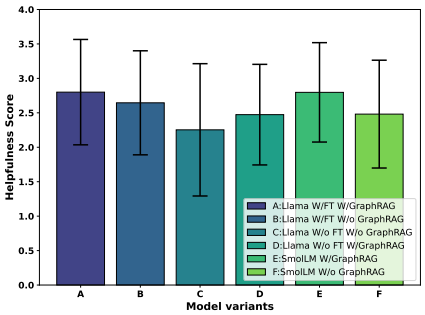


(e) Evaluation on test sets for SmolLM2-135M, fine-tuned via DPO on preference-aligned training splits derived from held-out data. Reported metrics include BLEU, ROUGE, Meteor, BERTScore, SacreBLEU, and Semantic Similarity.

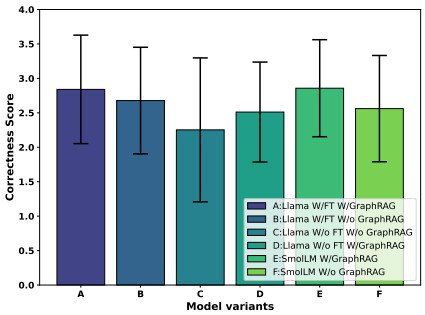


(f) Evaluation on test sets for SmolLM2-135M, fine-tuned on the training splits of multi-scale RAIT datasets (Local/Global). Metrics include BLEU, ROUGE, BERTScore, and Semantic Similarity.

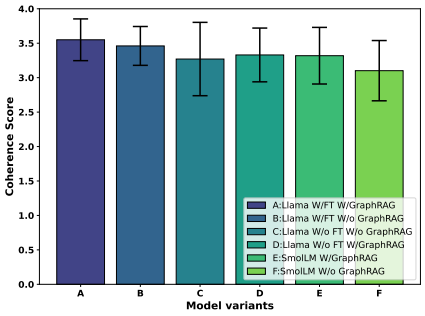
Figure 2: Evaluation on test sets for the Llama 3.2 1B and SmolLM2-135M models. Each model is fine-tuned on training splits using either supervised fine-tuning or fine-tuning via DPO (using preference-annotated data) with performance measured solely on the respective test sets.



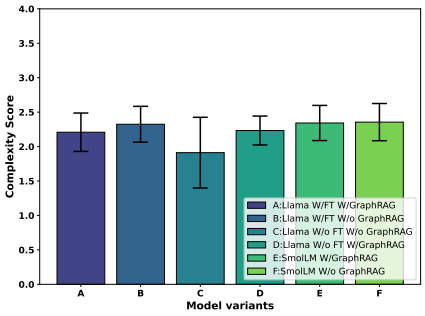
(a) The figure shows helpfulness scores on the test set. Fine-tuning and Graph RAG improve helpfulness, with Llama-3.2 1B variants outperforming SmolLM2-135M counterparts.



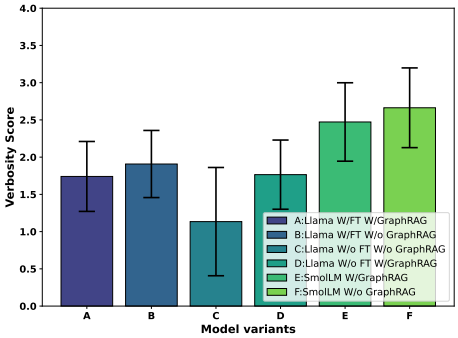
(b) The figure shows correctness scores on the test set. Retrieval augmentation contributes significantly to correctness, particularly for Llama-3.2 1B models.



(c) The figure illustrates coherence scores on the test set. Fine-tuning enhances structural fluency and logical flow, with Llama-3.2 1B models producing more coherent outputs.



(d) The figure shows complexity scores on the test set. Fine-tuned models generate more elaborate and informative responses, with Graph RAG aiding structured content generation.



(e) The figure shows verbosity scores on the test set. Fine-tuning increases verbosity, while Graph RAG helps regulate verbosity levels by encouraging more concise and content-rich responses.

Figure 3: The figure presents the reward-based evaluation of six model variants on a 1.5K QA-pair generalization benchmark, distinct from all synthetic training datasets used for model fine-tuning and evaluation (Factual QA, SynDIP, LogiCore, Local/Global RAIT, and DPO). Each model is scored using the Nvidia/Nemotron-4-340B reward model across five qualitative metrics: helpfulness, correctness, coherence, complexity, and verbosity. The variants differ by model size (Llama-3.2 1B vs. SmolLM2-135M), presence of fine-tuning, and usage of Graph RAG. The results highlight the complementary contributions of fine-tuning and retrieval to improving response quality across all metrics.