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

Sakhinana Sagar Srinivas*
sagar.sakhinana@tcs.com
Tata Research
Bangalore, India

Shivam Gupta*
shivam.gupta@tcs.com
Tata Research
New Delhi, India

Venkataramana Runkana
venkat.runkana@tcs.com
Tata Research
Pune, India

Abstract

Recent advancements in generative AI have accelerated the discovery of novel chemicals and materials, yet transitioning these discoveries to industrial-scale production remains a critical bottleneck. Current AI methods cannot auto-generate Process Flow Diagrams (PFDs) or Piping and Instrumentation Diagrams (PIDs)—essential for scaling chemical processes—while adhering to engineering constraints. We introduce a closed-loop, physics-aware framework that automates the generation of industrially viable PFDs and PIDs by integrating domain-specialized Small Language Models (SLMs) trained for chemical process QA tasks (e.g., equipment selection, control logic validation)with first-principles simulation validation. Our approach combines: (1) a hierarchical knowledge graph of 1,020+ chemicals for retrieval-augmented generation, (2) synthetic data pipelines fine-tuned via multi-stage training (Supervised Fine-Tuning (SFT), Direct Preference Optimization (DPO), and Retrieval-Augmented Instruction Tuning (RAIT)), and (3) simulator-in-the-loop validation (DWSIM) to ensure feasibility. Experiments demonstrate that the framework generates simulator-validated process descriptions with high fidelity, outperforms baseline methods in correctness and complexity, and generalizes to unseen chemicals. By bridging AI-driven design with industrial-scale feasibility, this work significantly reduces R&D timelines from lab discovery to plant deployment.

Keywords

Chemical Process Diagrams, Process Simulations, Agentic AI, Graph RAG, Pruning, KV Caching, Decoding, Test-Time Optimization

1 Introduction

Process Flow Diagrams (PFDs) and Piping and Instrumentation Diagrams (PIDs) are standard engineering diagrams in the chemical process industry. A PFD provides a high-level schematic (or layout) of the flow of materials and energy through a chemical production process. It depicts major equipment, process streams, and key operating conditions for specific equipment within the process, without delving into instrumentation or control details for plant operation. PIDs, which build upon PFDs, provide a more detailed schematic of the instrumentation and control systems that are essential for monitoring, operational control, safety, and maintenance of the chemical process plant. The purpose of a PFD(refer Figure 1) is to show what happens in the process (i.e., the key physical or chemical transformations) and where it happens (i.e., in which major equipment units), rather than how the process is controlled or executed. The purpose of a PID(refer Figure 2) is to illustrate how the process operates and is controlled (via piping, valves, sensors, and control loops), not just what happens or where it occurs. Recent advancements in generative AI are transforming chemical and materials science [7, 16, 20, 23, 34, 47, 55, 58, 58, 61] by accelerating the autonomous discovery of environmentally friendly, next-generation specialty chemicals and the development of low-cost, high-performance materials-based products. At the same time, these advancements are enhancing complex first-principles simulation workflows and reducing reliance on tedious lab trial-and-error, enabling faster innovation, lower R&D costs, and more sustainable chemical and material development. However, these approaches require the development of new production processes to transition novel discoveries from computer simulations or wet-lab experiments to industrial-scale production - a significant challenge in bringing better products to market faster. Current methods [2, 14, 19, 42, 48, 54] are not designed to auto-generate process schematics (e.g., PFDs) or instrumentation and control layouts (e.g., PIDs) for novel industrial-scale chemical production processes, significantly limiting their utility. Furthermore, these approaches overlook essential process context: for PFDs, this includes the high-level objectives—what the process achieves and in what sequence—while for PIDs, it requires details on how the process is operated, monitored, and controlled. As a result, they cannot justify design choices or the control and instrumentation logic needed for safe, stable, and efficient operations. Additionally, existing methods fail to integrate first-principles-based simulators to verify the physical and operational feasibility of generated process descriptions, further compromising industrial reliability. To address these limitations, we present a closed-loop, self-driving lab framework for the autogeneration of high-fidelity PFDs and PIDs, accelerating the development of novel chemical processes. Implemented as an enterprise-grade, cloud-based SaaS solution, our framework significantly expedites the simulation-to-lab-to-pilot-to-plant scale-up pipeline, ensuring that only industrially viable, sustainable, and efficient processes advance to commercialization. The platform functions as an end-to-end process modeling tool, automating design, simulation, and optimization with minimal human intervention. By integrating

*Both authors contributed equally to this research.

first-principles, physics-aware modeling with iterative reflection and adaptive learning, the framework continuously self-improves, thereby enhancing the reliability of AI-generated PFDs and PIDs. To address these challenges, we propose a framework combining three key innovations: (1) hierarchical knowledge graphs for retrieval-augmented generation, (2) domain-specialized SLMs fine-tuned via multi-stage training, and (3) physics-aware simulator validation. By integrating these components into a closed-loop system, our approach ensures industrial feasibility while maintaining generative flexibility. In the following section, we formalize this methodology, beginning with data curation and knowledge graph construction, followed by SLM training and simulation-based verification.

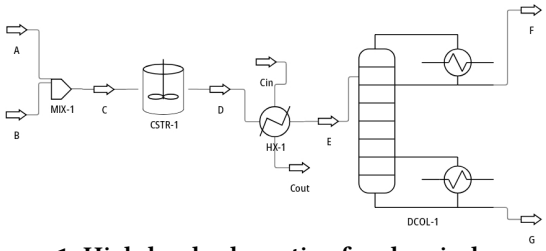

Figure 1: High-level schematic of a chemical process showing the material flow from reactant inlets (A and B) through a mixer (MIX-1), a continuous stirred-tank reactor (CSTR-1), a heat exchanger (HX-1), and a distillation column (DCOL-1) producing streams F and G. Major equipment and stream connections are shown, omitting instrumentation and control logic. This abstraction aids in understanding core process operations and transformations.

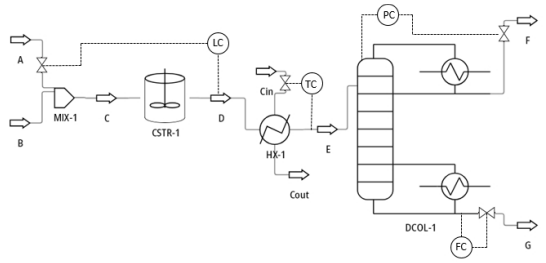

Figure 2: Piping and Instrumentation Diagram (P&ID). Detailed representation of a chemical process, including sensors, control valves, and feedback loops. The diagram shows level control (LC) on the reactor (CSTR-1) manipulating feed A, temperature control (TC) on the column feed (E) manipulating heat exchanger (HX-1) utility, pressure control (PC) at the top of the distillation column (DCOL-1) manipulating overhead product F, and flow control (FC) on the bottoms product G. This layout provides operational specificity essential for implementation, control strategy design, and safety analysis.

2 Methodology

Our methodology integrates data curation, knowledge graph construction, advanced language model fine-tuning, retrieval augmentation, inference optimization, and engineering validation to create specialized and efficient Small Language Models (SLMs) for chemical process engineering tasks, specifically PFD/PID interpretation, analysis, and generation. The pipeline begins with the curation of the ChemAtlas database (1,020+ common chemicals) from manufacturer catalogs and using an agentic web navigation framework that autonomously retrieves, interprets, and synthesizes multimodal data from diverse web sources to generate process flow and instrumentation descriptions of industrial production processes. This structured information is then used to construct a Knowledge Graph (KG), where text chunks are processed by GPT-4o to extract semantic triples (subject-predicate-object), entities are canonicalized based on high semantic (embedding) and string (Levenshtein) similarity, and the graph is partitioned into hierarchical communities using the Leiden algorithm to optimize modularity for efficient retrieval. Leveraging ChemAtlas, our curated database of 1,020 industrial chemicals, we employ advanced teacher LLMs (GPT-4o and Anthropic Claude Haiku) to generate and cross-validate over 20,000 synthetic QA pairs through self-instruct bootstrapping, initiated from a small seed set of human-written examples. These QA pairs are rigorously scored, validated, and filtered using NVIDIA's Nemotron-4-340B to ensure quality before training smaller language models (SLMs). The resulting datasets comprise six specialized subsets: (1) Factual QA for core process engineering fundamentals; (2) SynDIP datasets comprises comprehensive process contexts (reactions, operating conditions, control strategies), process flow and instrumentation descriptions(equivalent to agentic web retrieved knowledge on industrial chemical production but generated from LLMs pretrained knowledge); (3) LogiCore with multi-step reasoning chains for design justification, control logic validation and etc; (4) DPO containing preference-ranked pairs for alignment tuning across process engineering fundamentals; (5) Local RAIT (document-grounded QA using individual SynDIP technical documents), Global RAIT (cross-document synthesis QA requiring integration of multiple SynDIP sources) to enhance retrieval-augmented capabilities for complex engineering tasks. Additionally, we developed a 1.5K QA out-of-distribution benchmark using ChemAtlas to evaluate generalization performance on QA tasks, along with the ChemEval dataset (100 novel chemicals) to test zero-shot generation of complete process descriptions including process flow and instrumentation descriptions for unseen chemicals. Base SLMs, specifically Llama-3.2-1B and SmolLM-135M, are subsequently customized using QLoRA (Quantized Low-Rank Adaptation) using 4-bit NF4 precision with frozen base weights. Two primary fine-tuning strategies are employed on the synthetic datasets: first, a sequential pipeline involving Supervised Fine-Tuning (SFT) on Factual QA, SynDIP, and LogiCore datasets, followed by Direct Preference Optimization (DPO) on custom DPO datasets, and concluding with Retrieval-Augmented Instruction Tuning (RAIT) on Local/Global RAIT datasets; and second, a reinforcement learning approach adapting Group Relative Policy Optimization (GRPO), applied sequentially first

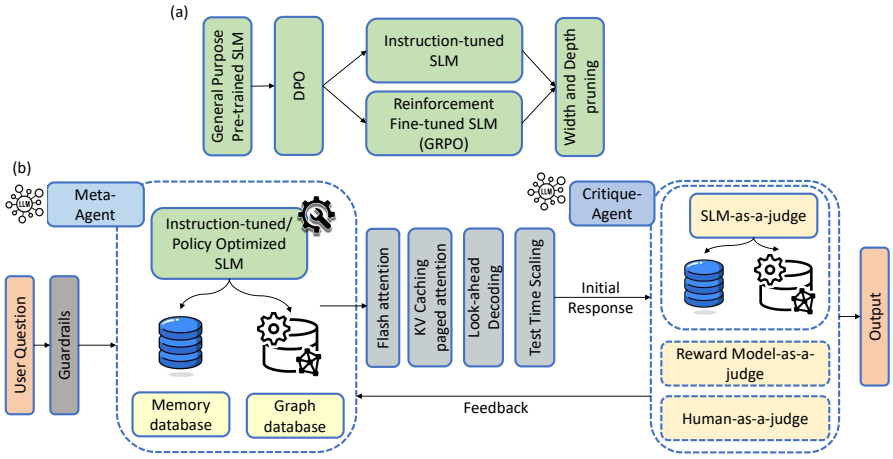

Figure 3: Overview of the integrated framework. (a) The SLM fine-tuning pipeline depicts initial DPO alignment followed by instruction tuning or GRPO reinforcement learning, concluding with optional width/depth pruning. (b) The operational RAG framework illustrates a Meta-Agent coordinating the specialized SLM (from part a), which accesses memory and graph databases for context. The SLM's inference is accelerated via optimizations (FlashAttention, Paged KV Caching, Lookahead Decoding, Test-Time Scaling). Generated responses are refined iteratively through a feedback loop managed by a Critique-Agent employing diverse judges (e.g., Nemotron-4-340B reward model, LLM-as-a-judge like GPT-4o/Haiku, or human evaluation).

on SFT datasets and then further refined on RAIT datasets, utilizing a composite reward function (combining ROUGE-L, length penalty, and LLM quality score) and KL divergence regularization for stability. The fine-tuned SLMs are integrated with the structured KG via Graph Retrieval-Augmented Generation (Graph RAG). During inference, relevant graph communities are retrieved based on query similarity to pre-computed summaries; a dynamic subgraph containing entities, relationships, and source chunks is constructed; and this context is passed to the SLM for grounded, multi-hop reasoning. To enhance performance, a suite of inference optimization and reliability techniques is implemented: structural pruning (width and depth) guided by importance heuristics reduces model size; PagedAttention combined with KV cache quantization mitigates memory fragmentation and reduces cache footprint; Lookahead Decoding accelerates generation latency through parallel token speculation; FlashAttention optimizes the core attention computation to reduce memory bandwidth bottlenecks; and Test-Time Inference Scaling improves output reliability using self-consistency sampling, confidence-weighted entropy scoring, iterative self-reflection/revision, and consensus aggregation. Finally, the practical engineering feasibility of generated PFD/PID descriptions is validated using the DWSIM open-source chemical process simulator, where PFDs are translated into flowsheets to verify material/energy balances and thermodynamic consistency, while PIDs are functionally assessed by implementing control loops in DWSIM's dynamic environment to evaluate stability and control performance (e.g., setpoint tracking, disturbance rejection). This comprehensive methodology ensures the developed SLMs are knowledgeable, aligned with engineering principles, efficient, and reliable for practical deployment. Figure 3 visually outlines the overall architecture. Part (a) depicts the SLM fine-tuning pipeline, showing the progression from a general pre-trained model through initial preference alignment (DPO), followed by task-specific fine-tuning via either instruction tuning or reinforcement learning (GRPO), and concluding with optional model compression (pruning). Part (b) illustrates the operational Retrieval-Augmented Generation (RAG) framework. It shows how a user query, after passing guardrails, is processed by a Meta-Agent. This agent utilizes the specialized SLM developed in part (a) as its core reasoning engine. The SLM, guided by the Meta-Agent, retrieves necessary context by accessing both a Memory database (e.g., for conversational history) and a Graph database (containing structured process knowledge). This retrieved information informs the SLM's response generation. The inference process of the SLM is further enhanced by integrated optimizations (FlashAttention, Paged Attention KV Caching, Lookahead Decoding, Test-Time Scaling). An initial response generated by the SLM undergoes evaluation by a Critique-Agent, utilizing feedback mechanisms (SLM-as-a-judge, Reward Model-as-a-judge, or Human-as-a-judge) to potentially trigger refinement before producing the final Output. In summary, our integrated framework leverages knowledge graph-based retrieval augmentation, domain-specific SLM fine-tuning pipelines, comprehensive inference optimizations, and feedback-driven refinement. This approach yields robust performance on complex reasoning tasks and demonstrates effective generalization via the generation of plausible, simulator-validated process descriptions for previously unseen chemicals.

3 Experiments

3.1 Datasets

In this work, we curated a chemical database comprising over 1,120 chemicals from sectors such as electronics, energy storage, pharmaceuticals, advanced manufacturing, and utilities, focusing on compounds critical to modern industrial and technological applications. The data were extracted from product catalogs of major manufacturers—including BASF, Dow Chemical, DuPont, Solvay, Mitsubishi Chemical, Bayer, Evonik, SABIC, and LyondellBasell—ensuring accuracy, reliability, and comprehensive coverage of industrial chemicals. The dataset consists of two components. The first, ChemAtlas, is a core collection of 1,020 chemicals. We employ 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 process flow and instrumentation descriptions of production processes. This structured information serves as the foundation for populating knowledge graphs, supporting reasoning and retrieval, and powers Graph RAG to enable small-scale language models (SLMs) to interpret context and answer user queries accurately. We then employ advanced large language models (LLMs), specifically GPT-4o and Anthropic Claude Haiku, to generate and cross-validate chemical-specific process flow and instrumentation descriptions obtained through agentic autonomous web navigation. These models leverage their pretrained knowledge for consistency checks. Additionally, we use a teacher-student transfer learning approach, generating custom synthetic datasets from the ChemAtlas database. This includes 20,000 question-answer (QA) pairs created by LLMs such as GPT-4o and Anthropic Haiku based on predefined templates 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 QA pairs through iterative synthesis. The textual outputs are scored, validated, and filtered using NVIDIA's Nemotron-4-340B to train smaller language models (SLMs)—such as Llama-3.2-1B and SmolLM-135M—for domain-specific tasks related to interpreting, analyzing, and auto-generating process flow and instrumentation descriptions. 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. Designed to enhance small-scale models' capabilities, these diverse datasets support tasks such as designing piping and equipment layouts, planning sensor and instrumentation layouts, and other related functions, enabling a deeper understanding of complex relationships within process engineering tasks. The curated collection consisted of six specialized synthetic subsets, each targeting a distinct model capability. The Factual QA dataset, generated via hierarchical topic decomposition, was curated to improve factual recall and foundational domain knowledge in process engineering. The SynDIP dataset comprised validated sequential trajectories linking process context, process flow, and instrumentation descriptions. In short, process context in a PFD describes what transformations occur, where they happen, and how material flows, while in a PID it defines how the system is monitored, controlled, and operated to ensure safe and efficient performance. The LogiCore dataset comprised multi-step reasoning QA pairs grounded in process flow and instrumentation descriptions, designed 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 a preferred and a dispreferred generated response, identified using reward model score differentials to enable alignment tuning via Direct Preference Optimization. The RAIT (Retrieval-Augmented Instruction Tuning) datasets were created to enhance the SLMs' ability to utilize retrieved information effectively. Local RAIT, by grounding QA pairs in individual text chunks from SynDIP documents, trains the model for precise, context-specific information extraction and answer generation. Global RAIT, by using context from semantically clustered groups of chunks (potentially spanning multiple documents), develops the model's capability to synthesize information across related segments, fostering cross-contextual reasoning and comprehension. In addition, we create a 1.5K QA-pair out-of-distribution (OOD) benchmark from ChemAtlas, using self-instruct with teacher LLMs (OpenAI o3 and o4-mini) to generate the synthetic QA pairs—iteratively from agentic web-retrieved information and filtered for quality—to test whether fine-tuned SLMs generalize across core capabilities, including factual knowledge, reasoning, instruction following, and process flows/instrumentation interpretation and analysis tasks. The second component, ChemEval, is a secondary evaluation set of 100 chemicals specifically curated to test the framework's robustness and generalization in auto-generating process flow and instrumentation descriptions for novel, unseen chemicals (for chemicals not present in ChemAtlas database). For each chemical, the teacher models (GPT-4o and Anthropic Haiku) generated process flow and instrumentation 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.

3.2 Experimental Setup

Graph Retrieval-Augmented Generation (Graph RAG) integrates structured knowledge graphs with large language models to enhance retrieval and reasoning. Our implementation begins with domain-specific documents—focused on chemical production processes—retrieved via autonomous agentic web navigation from the ChemAtlas database. The raw text is segmented into overlapping chunks using a sliding window approach, preserving local context while allowing cross-chunk continuity. Each text chunk is processed by GPT-4o to extract subject-predicate-object triples, forming semantic edges between entity nodes in the knowledge graph. Structural containment edges are simultaneously established, linking each entity back to its source chunk to maintain provenance. To

resolve redundancy, we apply a canonicalization step: entities are merged only if they exhibit both high semantic similarity (measured via text-embedding-3-small embeddings) and high string similarity (evaluated using normalized Levenshtein distance), with both metrics exceeding predefined thresholds. For efficient retrieval, we partition the graph into hierarchical communities using the Leiden algorithm[53], optimizing for modularity to ensure coherent topic-based clustering. During inference, each community is summarized by GPT-4o, and these summaries are embedded for fast similarity comparison. Given a query, the system retrieves the top-K most relevant communities, dynamically constructing a subgraph that includes their interconnected entities, semantic relationships, and originating chunks. This structured context is then passed to the reasoning model, ensuring grounded, multi-hop generation. We fine-tuned the Llama-3.2-1B and SmolLM2-135M models using Quantized Low-Rank Adaptation (QLoRA)[10], which adapts low-rank matrices for key transformer projection layers while keeping the base model weights frozen in 4-bit NormalFloat (NF4) precision. All experiments shared common training parameters: the 8-bit AdamW optimizer ($\beta_1 = 0.9, \beta_2 = 0.999$), a learning rate of 2×10^{-4} with linear decay, a weight decay of 0.01, an effective batch size of 8 (achieved via a per-device batch size of 2 and 4 gradient accumulation steps), and a maximum sequence length of 4096 tokens enabled by gradient checkpointing. Training was conducted using BF16/FP16 mixed precision on NVIDIA H100 GPUs. We explored two primary fine-tuning strategies. The first was a sequential multi-stage pipeline consisting of three phases: (1) Supervised Fine-Tuning (SFT) for 15 epochs on the combined Factual QA, SynDIP, and LogiCore datasets to establish domain knowledge and basic instruction-following capabilities; (2) Direct Preference Optimization (DPO) for 5 epochs using the curated DPO dataset to align the model with preferred response styles; and (3) Retrieval-Augmented Instruction Tuning (RAIT) for 15 epochs using the Local and Global RAIT datasets to enhance the model's ability to ground responses in retrieved contextual information. The second strategy adapted the Group Relative Policy Optimization (GRPO) reinforcement learning algorithm[44] for direct policy fine-tuning. We applied GRPO sequentially in two stages: first, fine-tuning the pretrained base model directly on the combined Factual QA, SynDIP, and LogiCore datasets; and second, further refining the resulting SFT checkpoint using the Local and Global RAIT datasets. In both stages, the policy (the small language model) was optimized under the same QLoRA setup described earlier. The optimization process maximized a clipped surrogate objective [41], analogous to Proximal Policy Optimization (PPO), leveraging normalized advantages derived from a composite reward function. This function aggregated three signals: the ROUGE-L F1 score (weight: 0.3), a length ratio penalty encouraging similarity to reference response lengths (weight: 0.2), and an auxiliary LLM-based quality score assessing correctness and relevance (weight: 0.5). Rewards and advantages were computed relative to groups of $G = 4$ responses sampled from the policy for each input. Training stability was maintained via β-weighted KL divergence regularization against the appropriate reference policy (either the pretrained base model or the SFT checkpoint). GRPO training proceeded for 10–15 epochs until convergence in both stages. All implementations were developed in PyTorch, utilizing libraries from the Hugging Face ecosystem, including `transformers`, `datasets`, `peft`, and `trl`. Model performance was comprehensively evaluated using quantitative textual metrics, qualitative reward model assessments, system-level efficiency benchmarks, and engineering process simulations. Quantitative analysis employed standard NLP metrics—including BLEU, ROUGE (1, 2, L), METEOR, SacreBLEU, BERTScore, and cosine similarity based on Sentence Transformer embeddings [39]—to assess textual fidelity and overlap. Qualitative aspects such as correctness, coherence, helpfulness, complexity, and verbosity were evaluated using the Nvidia Nemotron-4-340B reward model, which provided scores on a 0–4 scale for outputs generated on both the held-out test set and the ChemEval generalization benchmark. Additionally, the system-level efficiency impact of inference optimizations—PagedAttention [24] with KV cache quantization, Lookahead Decoding [12], and FlashAttention [8, 9]—was quantified through benchmarks measuring inference throughput (tokens/sec), average generation latency (seconds), maximum achievable batch size, and peak GPU memory usage (GB). We conducted a systematic evaluation of contemporary optimization techniques aimed at enhancing the efficiency and reliability of fine-tuned Small Language Models (SLMs). Leveraging the Llama-3.2-1B architecture on NVIDIA H100 GPUs, our investigation quantified the individual contributions of several methods: structural pruning for model compression, PagedAttention for memory-efficient inference, Lookahead Decoding and FlashAttention for latency reduction via distinct mechanisms, and test-time inference scaling for improving output reliability without additional retraining. This analysis aimed to elucidate both the standalone and synergistic benefits of these complementary strategies for demanding engineering applications. To examine the trade-off between model compression and predictive fidelity, we applied structural pruning techniques. Both width-level (neuron-level) and depth-level (layer-level) pruning were guided by importance heuristics computed during fine-tuning, enabling systematic reduction of model parameters while analyzing the impact on downstream performance. To enhance inference-time reliability—particularly in terms of factual accuracy and reasoning consistency—we implemented a test-time scaling mechanism. This approach employs multi-path exploration via stochastic sampling, confidence-based candidate ranking, iterative self-reflection and revision, and consensus aggregation. Collectively, these techniques improve output robustness relative to standard deterministic decoding, as measured by qualitative reward model metrics. PagedAttention was evaluated to address memory fragmentation and enhance throughput during inference. By organizing the Key-Value cache into non-contiguous memory blocks, this approach improves memory efficiency

and enables larger batch sizes compared to conventional contiguous caching strategies. Its performance was assessed using standard system-level metrics. We applied Lookahead Decoding to accelerate autoregressive generation by reducing end-to-end latency. This technique allows for parallel token generation and verification within each forward pass while preserving output equivalence with greedy decoding. Effectiveness was measured by comparing generation latency and throughput against baseline decoding methods. Lastly, we integrated FlashAttention to accelerate attention computation and alleviate memory bandwidth bottlenecks inherent to Transformer architectures. As an I/O-aware optimization, FlashAttention improves efficiency within the attention mechanism. We evaluated its impact on both training and inference throughput, as well as peak memory consumption during training. Finally, engineering feasibility was validated using the DWSIM open-source chemical process simulator [32]. Textual PFD descriptions generated by the framework were manually translated into DWSIM flowsheets to verify material and energy balances, as well as thermodynamic consistency, via simulation. Correspondingly, control strategies derived from textual PID descriptions were functionally assessed by implementing control loops in DWSIM's dynamic simulation environment and evaluating closed-loop stability and performance (e.g., setpoint tracking and disturbance rejection).

3.3 Results

We present a comparative evaluation of Llama-3.2-1B and SmolLM-135M across successive fine-tuning stages, using held-out test splits and a comprehensive set of performance metrics. Evaluation was conducted using both token-level n-gram overlap metrics (BLEU, ROUGE-1/2/L, METEOR, SacreBLEU) and embedding-based semantic similarity metrics (BERTScore and Sentence-BERT cosine similarity), with all scores normalized to the $[0, 1]$ interval, as shown in Figures 4a–4f. Following supervised fine-tuning (SFT) on the train splits of the Factual QA, SynDIP, and LogiCore datasets, Llama-3.2-1B (Figure 4a) demonstrates strong semantic alignment. This is evidenced by high BERTScore and sentence similarity, despite lower performance on n-gram metrics, indicating a preference for paraphrastic generation over lexical overlap. In contrast, SmolLM-135M (Figure 4d), fine-tuned on the same training data, exhibits relatively higher n-gram overlap scores and sentence similarity, while achieving moderate BERTScore, suggesting a tendency toward surface-level fidelity. Subsequent to Direct Preference Optimization (DPO), trained on the train split of the DPO dataset and evaluated on its test split, Llama-3.2-1B (Figure 4b) maintains its profile of high semantic similarity, whereas SmolLM-135M (Figure 4e) displays balanced improvements across both lexical and semantic metrics, reflecting effective alignment tuning via reward modeling. Following Retrieval-Augmented Instruction Tuning (RAIT), which was performed using the train splits of the Local and Global RAIT datasets and evaluated on their respective test splits, Llama-3.2-1B (Figure 4c) continues to show dominant semantic scores relative to n-gram metrics. SmolLM-135M (Figure 4f) exhibits comparatively lower scores across most metrics, with sentence similarity remaining the strongest signal, suggesting diminished generalization capacity under retrieval-augmented long-context settings. These plots provide phase-by-phase performance diagnostics, highlighting how successive fine-tuning regimes induce distinct response behaviors across models in terms of semantic coherence, lexical fidelity, and alignment with training objectives. In addition, we conduct a systematic evaluation of how fine-tuning (FT) and Graph Retrieval-Augmented Generation (Graph RAG) affect qualitative performance across six language model variants, comprising two model architectures at different scales: the larger Llama-3.2-1B and more compact SmolLM2-135M. Each model variant represents a distinct configuration: (a) Llama-3.2-1B with both FT and Graph RAG (W/FT W/Graph RAG), (b) Llama-3.2-1B with FT but without Graph RAG (W/FT W/o Graph RAG), (c) Llama-3.2-1B without either FT or Graph RAG (W/o FT W/o Graph RAG), (d) Llama-3.2-1B without FT but with Graph RAG (W/o FT W/Graph RAG), (e) SmolLM2-135M with both FT and Graph RAG (W/FT W/Graph RAG), and (f) SmolLM2-135M with FT but without Graph RAG (W/FT W/o Graph RAG). Performance is rigorously assessed using the NVIDIA Nemotron-4-340B reward model across five key qualitative dimensions: helpfulness (measuring practical utility), correctness (factual accuracy), coherence (logical flow), complexity (depth of content), and verbosity (response length), with detailed results presented in Figures 5a-e. The evaluation reveals several important findings regarding model scale and methodological impact. Among Llama-3.2-1B variants, the FT+RAG configuration (variant a) demonstrates superior performance, achieving the highest scores in correctness and complexity by effectively combining fine-tuned capabilities with retrieved knowledge, though this comes with increased verbosity due to the incorporation of supplementary content from the knowledge graph. The FT-only variant (b) maintains strong performance in coherence and helpfulness but shows limitations in knowledge-intensive tasks without retrieval support. Notably, even without fine-tuning, the Graph RAG-enabled Llama variant (d) outperforms the baseline (c) in correctness, demonstrating that retrieval augmentation can partially compensate for the absence of task-specific tuning. However, the complete absence of both methods (variant c) results in the weakest performance, highlighting the limitations of relying solely on pretrained knowledge. For the smaller SmolLM2-135M models, Graph RAG improves correctness (variant e vs. f), but both configurations underperform relative to the corresponding Llama-3.2-1B variants across all metrics, notably in coherence and complexity. This performance gap underscores the importance of model scale in effectively utilizing both fine-tuning and retrieval augmentation. The results demonstrate that while FT substantially enhances overall response quality across all metrics by aligning the model with domain-specific requirements, Graph RAG provides complementary benefits primarily for factual accuracy. This combination proves particularly valuable in specialized domains like chemical process

synthesis, where both task adaptation and external knowledge integration are crucial for high-quality generation. The study conclusively shows that the optimal configuration—Llama-3.2-1B with both FT and Graph RAG—achieves the most balanced performance across all evaluation dimensions, successfully integrating structured knowledge retrieval with fine-tuned language understanding capabilities while maintaining reasonable verbosity levels. These findings have important implications for deploying language models in technical domains where both factual precision and contextual understanding are paramount.

4 Conclusion

Automating the generation of industrially viable Process Flow Diagrams (PFDs) and Piping and Instrumentation Diagrams (PIDs) is critical for accelerating chemical process scale-up. Our closed-loop framework achieves this by integrating domain-adapted small language models (SLMs) with physics-aware validation to enable end-to-end automation. The approach combines multi-stage SLM fine-tuning—leveraging synthetic datasets and retrieval augmentation from a hierarchical chemical knowledge graph—with rigorous simulation-based validation using DWSIM. Results demonstrate that the synergy between fine-tuning and Graph Retrieval-Augmented Generation (RAG) enables high-fidelity PFD/PID generation with strong generalization capabilities. Specifically, the framework exhibits robust performance in zero-shot synthesis of novel chemical production processes and excels at core engineering QA tasks, including PFD/PID interpretation and analysis. By unifying generative AI with first-principles engineering constraints, the framework effectively bridges the gap between digital discovery and industrial deployment, addressing key R&D bottlenecks. Future work will focus on expanding capabilities through deeper integration with closed-loop simulation for automated process optimization and improving usability through direct generation of standard-format engineering diagrams (e.g., Visio, CAD). This work presents a validated pathway toward more efficient and reliable AI-driven chemical process design.

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

[11] 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).

[12] 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).

[13] 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).

[14] 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).

[15] 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).

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

[17] 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).

[18] 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.

[19] 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).

[20] 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*.

[21] 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).

[22] 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).

[23] 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).

[24] 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.

[25] 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).

[26] 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).

[27] 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).

[28] 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).

[29] 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).

[30] 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.

[31] 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).

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

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

[34] 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*.

[35] 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).

[36] 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).

[37] 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).

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

[39] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084* (2019).

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

[41] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[42] 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.

[43] 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.

[44] 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).

[45] 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).

[46] 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).

[47] 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).

[48] Sakhinana Sagar Srinivas, Akash Das, Shivam Gupta, and Venkataramana Runkana. 2024. Accelerating Manufacturing Scale-Up from Material Discovery Using Agentic Web Navigation and Retrieval-Augmented AI for Process Engineering Schematics Design. *arXiv preprint arXiv:2412.05937* (2024).

[49] 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).

[50] 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).

[51] 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).

[52] 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.

[53] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. 2019. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific reports* 9, 1 (2019), 1–12.

[54] 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.

[55] 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).

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

[57] 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).

[58] 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.

[59] 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).

[60] 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).

[61] 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).

[62] 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).

[63] 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.

[64] 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).

[65] 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.