Large Language Model for Autogeneration of Engineering Diagrams

Anonymous Author(s)

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.

CCS Concepts

• **Do Not Use This Code → Generate the Correct Terms for Your Paper**; *Generate the Correct Terms for Your Paper*; Generate the Correct Terms for Your Paper; Generate the Correct Terms for Your Paper.

Keywords

Do, Not, Us, This, Code, Put, the, Correct, Terms, for, Your, Paper

ACM Reference Format:
Anonymous Author(s). 2018. Large Language Model for Autogeneration of Engineering Diagrams. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 24 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

ACM's consolidated article template, introduced in 2017, provides a consistent LaTeX style for use across ACM publications, and incorporates accessibility and metadata-extraction functionality necessary for future Digital Library endeavors. Numerous ACM and SIG-specific LaTeX templates have been examined, and their unique features incorporated into this single new template.

If you are new to publishing with ACM, this document is a valuable guide to the process of preparing your work for publication.

2 Graph RAG

The Retrieval-Augmented Generation (RAG) approach utilizes external knowledge databases to assist large language models (LLMs), enabling them to effectively retrieve isolated facts for in-domain question-answering tasks. Graph RAG enhances traditional RAG by incorporating structured knowledge graphs (relational graph data) for knowledge retrieval. This integration helps address complex, multi-step reasoning tasks by synthesizing information from multiple, disparate sources. It facilitates the traversal of multi-hop relationships, improving contextual understanding and reasoning processes, which leads to more nuanced, detailed, and accurate responses in open-domain question-answering (ODQA) tasks. We use specialized agents for autonomous web navigation to gather chemical-specific multimodal data from various online sources to find and collect specific information related to PFDs and PIDs. This aggregated web data is stored as documents and is further used to populate knowledge graphs. The process begins by constructing ontological graphs from unstructured documents, where relevant data is extracted and organized/indexed into a graph database. To process the document t_i, we break the extracted text into smaller chunks using a sliding window technique. Let $C_i = \{c_1, c_2, \ldots, c_M\}$ represent the set of text chunks extracted from t_i, where each chunk c_j is a segment of size $|c_j|$. We define a window size w and a stride s, and apply the sliding window operation such that each chunk c_j is a subsegment of text from position p_j to $p_j + w - 1$, where $p_j = 1 + (j-1) \cdot s$. This method produces overlapping chunks, ensuring contextual continuity. Each chunk c_j is treated as a chunk node in the knowledge graph. To further enrich the chunks, we use a language model $\mathcal{M}_\theta$ to generate a semantic description $\mathcal{R}_j$ that explains the relationship between chunk c_j and other chunks in C_i. The relational description is generated by the language model as:

$$\mathcal{R}_j = \mathcal{M}_\theta(c_j, C_i \setminus \{c_j\}) \tag{1}$$

This description is then appended to c_j, forming the updated chunk $c'_j = c_j \oplus \mathcal{R}_j$. This process results in a set of augmented chunks, each enriched with contextual information, thereby improving their representation for tasks like graph-based retrieval and generation. Each augmented chunk c'_j contains entities (subjects and objects) and relations (predicates). The goal is to extract triples from these chunks, where each triple represents a subject-predicate-object relationship. For a given chunk c'_j, the following extraction process occurs: (1) The entities in the chunk are represented as entity nodes (distinct from the former chunk nodes). We denote the set of entities in chunk c'_j as $E_j = \{e_{j1}, e_{j2}, \ldots, e_{jK_j}\}$, where e_{jk} represents the k-th entity in chunk c'_j. Here, K_j represents the cardinality of the set E_j, which is the total number of entities in chunk c'_j. (2) The relations between entities are represented as edges. We denote the set of relations (predicates) in chunk c'_j as $R_j = \{r_{jkm} \mid 1 \leq k \neq m \leq K_j\}$, where r_{jkm} represents the relation between entities e_{jk} and e_{jm} in chunk c'_j. Thus, the triple extraction process for chunk c'_j can be written as:

$$\mathcal{T}_j = \{(e_{jk}, r_{jkm}, e_{jm}) \mid 1 \leq k \neq m \leq K_j\} \tag{2}$$

where each triple $(e_{jk}, r_{jkm}, e_{jm})$ represents a relation between entities (e_{jk}) and (e_{jm}), connected by the relation (r_{jkm}) in chunk (c'_j). The set of extracted triples from all augmented chunks ($C'_i = \{c'_1, c'_2, \ldots, c'_M\}$) forms the basis of the knowledge graph $(\mathcal{G}_i)$, where the extracted triples provide directed edges between entities (subjects and objects) connected by predicates (relations). Furthermore, each entity (e_{jk}) is associated with the chunk node (c'_j), which represents the chunk from which the entity was extracted. These associations can be formalized as:

$$(e_{jk}, c'_j) \in R_j, \quad (e_{jl}, c'_j) \in R_j \tag{3}$$

where (R_j) now refers to the set of relations (including both entity-to-entity relations and chunk-node associations) for chunk (c'_j), indicating the connection between the chunk node (c'_j) and its associated entities. The final knowledge graph $(\mathcal{G}_i)$ can be formally defined as a directed graph ($\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$), where the set of nodes $(\mathcal{V}_i)$ consists of both chunk nodes and entity nodes. Specifically, the set of nodes is:

$$\mathcal{V}_i = \left\{c'_1, c'_2, \ldots, c'_M\right\} \cup \left\{e_{jk} \mid j = 1, 2, \ldots, M \quad \text{and} \quad k = 1, 2, \ldots, K_j\right\} \tag{4}$$

This set of nodes includes chunk nodes (c'_j) as well as entity nodes (e_{jk}) extracted from the chunks. The edges $(\mathcal{E}_i)$ in the graph consist of two types. The first type is relation edges, which represent the subject-predicate-object relationships between entities, and are defined as:

$$\mathcal{E}_i^{\text{rel}} = \left\{(e_{jk}, r_{jkm}, e_{jm}) \mid j = 1, 2, \ldots, M \quad \text{and} \quad 1 \leq k \neq m \leq K_j\right\} \tag{5}$$

The second type is containment edges, which represent the relationship where a chunk (c'_j) contains an entity (e_{jk}), and are defined as:

$$\mathcal{E}_i^{\text{cont}} = \left\{(c'_j, e_{jk}) \mid j = 1, 2, \ldots, M \quad \text{and} \quad k = 1, 2, \ldots, K_j\right\} \tag{6}$$

The total set of edges $(\mathcal{E}_i)$ is then the union of both relation and containment edges:

$$\mathcal{E}_i = \mathcal{E}_i^{\text{rel}} \cup \mathcal{E}_i^{\text{cont}} \tag{7}$$

In short, $\mathcal{G}_i$, a heterogeneous, directed, multi-dimensional graph structure with chunk nodes, entity nodes, and relation and containment edges, represents both semantic relationships between entities and structural relationships between entities and chunks, enabling graph-based retrieval, reasoning, and generation. We improve knowledge retrieval accuracy by identifying and merging duplicate entities that represent the same concept in different ways. Each entity is encoded as a vector embedding v_{jk}, which is generated using the text-embedding model to capture its semantic representation. To determine if two entities (e_{jk}) from chunk (c'_j) and $(e_{j'k'})$ from chunk $(c'_{j'})$ refer to the same concept, we first compute the cosine similarity between their vector embeddings:

$$\texttt{sim}(v_{jk}, v_{j'k'}) = \frac{v_{jk} \cdot v_{j'k'}}{\|v_{jk}\| \|v_{j'k'}\|} \tag{8}$$

If this similarity score exceeds a predefined threshold (τ_{sim}), we then calculate the string-based similarity using the Levenshtein distance, which quantifies how many character edits are needed to transform one string into another:

$$\texttt{str\_sim}(e_{jk}, e_{j'k'}) = 1 - \frac{d_{\text{lev}}(e_{jk}, e_{j'k'})}{\max(|e_{jk}|, |e_{j'k'}|)} \tag{9}$$

where $d_{\text{lev}}(e_{jk}, e_{j'k'})$ is the Levenshtein distance between these two strings and $|e_{jk}|$ and $|e_{j'k'}|$ denote the lengths of the strings (e_{jk}) and $(e_{j'k'})$, respectively. Entities (e_{jk}) from chunk (c'_j) and $(e_{j'k'})$ from chunk $(c'_{j'})$ are considered duplicates and merged if both the semantic similarity and the string similarity surpass their respective thresholds (τ_{sim}) and (τ_{str}). We apply the hierarchical Leiden algorithm to detect communities (C_k) at various granularities within the knowledge graph ($\mathcal{G}_i$), aiming to optimize modularity (M_{Mod}). Modularity measures the quality of the community structure by comparing the density of connections within communities to those between them. It is defined as:

$$M_{\text{Mod}} = \frac{1}{2n}\sum_{i,j}\left[\mathcal{A}_{ij} - \frac{d_i d_j}{2n}\right]\delta(c_i, c_j) \tag{10}$$

where ($\mathcal{A}_{ij}$) is the adjacency matrix, indicating the presence of an edge between nodes (i) and (j), and d_i and d_j are the degrees of nodes (i) and (j). (n) is the total number of edges, normalizing the adjacency matrix to reflect expected density in a random graph where node degrees are preserved but connections are randomized. ($\delta(c_i, c_j)$) is the Kronecker delta function, equal to 1 if nodes (i) and (j) belong to the same community, and 0 otherwise. A community ($C_k = (\mathcal{V}_{C_k}, \mathcal{E}_{C_k})$) is a subset of nodes ($\mathcal{V}_{C_k}$) (entities) and edges ($\mathcal{E}_{C_k}$) (relationships) in the knowledge graph that are more densely connected internally than externally. These communities help optimize the graph for tasks like retrieval and reasoning. The hierarchical Leiden algorithm partitions the knowledge graph ($\mathcal{G}_k$) into (L) communities ($\{C_1, C_2, \ldots, C_L\}$), maximizing ($M_{\text{Mod}}$) and revealing a strong community structure. The process consists of three steps: (1) Local Moving Phase, where the nodes and edges are reassigned between communities to maximize modularity. (2) Aggregation Phase, where communities are merged into super-nodes, which represent collections of entities and their relationships. (3) Repetition, where the process is repeated until no further improvement in modularity is achieved.

For complex reasoning tasks, relevant information often spans across multiple communities, there is a need and necessity for efficient retrieval within the knowledge graph by detecting and retrieving communities aligned with query-specific subgraphs. To address this, we rank the top-K communities $\{C_1, C_2, \ldots, C_K\}$ based on their cosine similarity to the user query (Q) and the summaries of relationship paths within each community. Each community (C_k) is summarized using a language model ($\mathcal{M}_\theta$), which condenses the relationship paths (Rel_k):

$$s_k = \mathcal{M}_\theta(\text{Rel}_k) = \arg\max_{S} P(S \mid \text{Rel}_k) \tag{11}$$

Where (s_k) represents the summary, and ($P(S \mid \text{Rel}_k)$) is the likelihood of (S) given (Rel_k). These summaries are transformed into vector embeddings ($v(S_k)$) using a text-embedding model, enabling efficient similarity computation with the query embedding ($v(Q)$):

$$d(Q, C_k) = \frac{\langle v(Q), v(S_k)\rangle}{\|v(Q)\|\|v(S_k)\|} \tag{12}$$

The top-K communities with the highest similarity scores are selected. These communities are then combined into a query-specific subgraph ($\mathcal{G}_Q = (V_Q, E_Q)$), defined as:

$$V_Q = \bigcup_{k=1}^{K} V_{C_k}, \quad E_Q = \bigcup_{k=1}^{K} E_{C_k} \tag{13}$$

This subgraph captures critical nodes (V_Q) and edges (E_Q) from the top-K communities, retaining the essential relationships needed to address the user query. Paths ($\mathcal{P} = (e_1, r_1, e_2, \ldots, e_m)$), where ($e_i$) are entities and ($r_i$) are relationships, are extracted from ($\mathcal{G}_Q$). These paths encode semantic dependencies that are vital for reasoning. A language model ($\mathcal{M}_\theta$) then generates the answer ($\mathcal{A}$) by incorporating the query (Q) and paths ($\mathcal{P}$):

$$\mathcal{A} = \mathcal{M}_\theta(Q, \mathcal{P}) = \arg\max_{\mathcal{A}} P(\mathcal{A} \mid Q, \mathcal{P}) \tag{14}$$

Where ($P(\mathcal{A} \mid Q, \mathcal{P})$) represents the probability of answer ($\mathcal{A}$) given the query (Q) and paths ($\mathcal{P}$), ensuring that the response aligns with the query's intent.

2.1 Datasets

In this work, we curated a chemical database of over 1,120 chemicals spanning key sectors such as electronics, energy, and pharmaceuticals, focusing on those critical to modern manufacturing and utilities. The data was extracted from product catalogs of major manufacturers, including BASF, Dow Chemical, DuPont, Solvay, Mitsubishi Chemical, Bayer, Evonik, SABIC, and LyondellBasell. This approach ensures high data fidelity, reliability, and comprehensive coverage of industrial chemicals, enhancing the dataset's relevance and practical utility. The dataset comprises two parts: *ChemAtlas*, a primary set of 1,020 chemicals designed for knowledge-generation tasks, such as prompting advanced LLMs like GPT-4o and Anthropic Haiku to generate chemical-specific PFDs and PIDs by leveraging their extensive pre-trained knowledge. Additionally, we independently utilize an autonomous agentic web navigation framework to produce detailed PFD and PID descriptions, thereby constructing ontological knowledge graphs as foundational databases. The second part, *ChemEval*, is a secondary evaluation set of 100 chemicals specifically curated to test the framework's robustness and generalization in auto-generating PFDs and PIDs. This work also employs a teacher-student transfer learning approach, generating custom synthetic ODQA datasets with 20,000 QA pairs using large language models (LLMs) such as GPT-4o and Anthropic Haiku. The textual outputs are scored, validated, and filtered using Nvidia-nemotron-4-340B to train smaller language models (e.g., Llama-3.2-1B, SmolLM-135M) for domain-specific tasks related to PFDs and PIDs. These datasets consist of QA pairs focusing on factual knowledge, preference optimization, chemical process synthesis descriptions, and logical reasoning. Specifically, the methodology involves using LLMs like GPT-4o and Claude-3-Haiku to generate questions based on predefined templates and corresponding answers, which are then evaluated using a reward model (Nvidia-nemotron-4-340B) for helpfulness, correctness, and coherence. Various structured workflows were employed to create different datasets, including a closed-ended factual QA dataset, a Direct Preference Optimization (DPO) dataset, a dataset of chemical process synthesis descriptions (SynDIP), a reasoning-focused dataset (LogiCore), and two Retrieval-Augmented Instruction-Tuning (RAIT) datasets (Local and Global) based on parsed PDF documents. These diverse datasets aim to enhance small-scale models' ability to interpret process flows, identify bottlenecks, optimize operations, and understand complex relationships within process engineering domains.

2.2 Experimental Settings

We fine-tuned the Llama 3.2 1B model with QLoRA (Quantized Low-Rank Adaptation) for task-specific applications. The pre-trained weights were frozen and quantized to 4-bit precision, reducing memory usage while supporting long input sequences. Additional low-rank matrices in full precision (e.g., FP16) were introduced to key layers such as query, key-value, output, and gate projections, with a rank of 16 and a scaling factor of 16. During forward computation, the 4-bit weights were combined with LoRA weights, and only the LoRA weights were updated during backward passes, ensuring efficient fine-tuning on resource-constrained hardware without compromising performance. Key training parameters included a batch size of 2 per GPU (the workload for each GPU in a distributed training setup), with gradient accumulation over 4 steps to simulate an effective batch size of 8. This approach allowed gradients to be calculated and accumulated over multiple forward and backward passes before updating the LoRA weights, optimizing memory usage and maintaining training stability. The training process began with 10 warm-up steps, during which the learning rate (lr) was gradually increased from zero to its target value ($lr = 1 \times 10^{-3}$), stabilizing updates and preventing abrupt optimization changes. Training concluded after 60 optimization steps, where accumulated gradients were used to iteratively update the model's trainable parameters, ensuring convergence and effective learning rather than being constrained by a fixed number of epochs. The AdamW optimizer used block-wise quantization to store parameters such as gradients, momentum, and variance in 8-bit precision, significantly reducing memory usage while maintaining performance comparable to 32-bit optimizers. A weight decay rate of 0.01 was applied to prevent overfitting, and the learning rate was dynamically adjusted using a linear scheduler for stable convergence. Mixed-precision training techniques, specifically BF16 (16-bit brain floating point), were optimized for NVIDIA A100 GPUs to accelerate computation and ensure training stability. Random seeds ensured reproducibility, and all training outputs were logged for analysis.

2.3 •

We utilize DWSIM, an advanced open-source chemical process simulator, for modeling material and energy balances in chemical process plants under steady-state and dynamic conditions. It is a comprehensive platform for design, engineering, and debottlenecking. DWSIM enhances our framework by enabling practical simulation and validation of autogenerated textual descriptions of PFDs and PIDs. As a simulator, DWSIM complements this work by simulating described processes in greater detail, allowing users to validate and refine textual descriptions with quantitative data. Its primary role is as a secondary tool for deeper process validation and optimization, bridging the gap between high-level schematic overviews and detailed engineering simulations. We use it to design, configure, and analyze PFDs and PIDs, leveraging a library of unit operations such as mixers, separators, reactors, distillation columns, and custom operations. The simulator includes robust thermodynamic and physical property packages for precise material and energy stream modeling, along with tools for reaction modeling, property estimation, sensitivity studies, and process optimization. It also includes cost estimation, greenhouse gas emissions modeling, and particle size distribution management for multiphase processes. Single-object calculations allow for sensitivity analysis without full flowsheet recalculations. A bidirectional solver facilitates forward and backward flow data propagation, enabling reverse engineering of input parameters for complex, looped, or recycled processes based on desired outputs.

2.4

The agentic web navigation framework is an advanced autonomous system designed to navigate the web for information retrieval, gather domain-specific knowledge, and synthesize insights for generating Process Flow Diagrams (PFDs) and Piping and Instrumentation Diagrams (PIDs) for large-scale industrial chemical synthesis. It produces detailed textual outputs, including process descriptions, equipment specifications, and instrumentation details, aligned with industrial standards. At its core, a meta-agent orchestrates the workflow by decomposing a primary query Q into subtasks $\{q_1, q_2, \ldots, q_n\}$, assigning them to specialized expert agents—such as the Visual Miner Agent, the Research Agent, the Patent Agent, and the Wiki Agent—each optimized for specific data retrieval tasks using tools like SerpAPI. The Visual Miner Agent analyzes diagrams to generate contextual summaries, while the Research, Patent, and Wiki Agents extract and synthesize knowledge from peer-reviewed articles, patent databases, and Wikipedia to provide accurate and comprehensive information for industrial chemical synthesis. For each subtask q_i, the optimal expert agent t_j^* is selected as:

$$t_j^* = \arg\max_j \text{sim}_{\cos}(v(q_i), v(d_j)), \quad (15)$$

where the similarity between a query q_i and an expert agent's capabilities d_j is defined as:

$$\text{sim}_{\cos}(v(q_i), v(d_j)) = \frac{v(q_i) \cdot v(d_j)}{\|v(q_i)\| \cdot \|v(d_j)\|}. \quad (16)$$

Here, $v(q_i)$ is the vector embedding of the subtask query, and $v(d_j)$ is the embedding of each expert agent's capabilities, defined by their data type (visual, academic, patent, or wiki), tools like SerpAPI, and expertise in information extraction. Subtasks $(q_1, q_2, \ldots, q_n)$ are represented as nodes $(v_1, v_2, \ldots, v_n)$ in a Directed Acyclic Graph (DAG), $G = (V, E)$, where each v_i corresponds to q_i. Edges (E) define precedence constraints, with e_{ij} indicating that v_j must be completed before v_i. It ensures dependencies are respected while enabling the parallel execution of independent tasks for efficient workflow management. Each expert agent generates a set of candidate results $M = \{m_1, m_2, \ldots, m_k\}$, where k represents the total number of retrieved candidates ($|M|$). The relevance of each result m_i is evaluated against the query embedding $v(q_i)$ using cosine similarity:

$$\text{sim}_{\cos}(v(m_i), v(q_i)) = \frac{v(m_i) \cdot v(q_i)}{\|v(m_i)\| \cdot \|v(q_i)\|}. \quad (17)$$

The top-K candidates, where $K \leq k$, are selected based on this metric. The agent synthesizes these top-ranked results into a single, comprehensive output tailored to the subtask q_i. Results from the expert agents are aggregated into a coherent response $A = \mathcal{F}_{\text{Meta}}(\{R_{q_i}\})$, which undergoes iterative refinement $A_{i+1} = \mathcal{F}_{\text{Meta}}(A_i, F_i)$ using feedback from human experts, AI judges, and reward models. AI judges act as critiques, qualitatively evaluating the correctness, relevance, and coherence of the outputs, while reward models assign a quantitative score based on alignment with desired objectives. Together, these mechanisms ensure high-quality results. In summary, the framework employs similarity computation for robust expert-agent selection, a DAG-based task prioritization method, and multimodal embedding techniques to enhance retrieval precision and contextual coherence. This structured and iterative approach optimizes knowledge generation for complex queries, facilitating the automated creation of regulation-compliant PFDs and PIDs.

2.5 Datasets

In this work, we employ teacher-student transfer learning, leveraging large-scale language models (LLMs) to generate synthetic QA pair datasets. These datasets are used to customize small-scale models with domain-specific expertise, facilitating knowledge transfer for tasks such as interpreting, analyzing, and generating Process Flow Diagrams (PFDs) and Process Instrumentation Diagrams (PIDs). Fine-tuning small-scale models on these synthetic datasets enhances their ability to effectively address end-user queries. This capability enables small-scale models to identify bottlenecks and inefficiencies in material flows and energy usage, optimize equipment layouts and operations, and recommend process adjustments to improve safety and ensure regulatory compliance. Figure 1 outlines a structured approach to synthetic data generation for closed-ended, domain-specific factual QA datasets using GPT-4o and Nvidia-nemotron-4-340B. The process begins with generating sub-topics from a given topic (e.g., PFDs, PIDs), followed by crafting multiple questions per sub-topic and producing corresponding responses. These responses are scored on dimensions such as helpfulness, correctness, coherence, complexity, and verbosity using a reward model. Responses that exceed a predefined threshold are filtered and saved, ensuring high-quality datasets for QA tasks. This process emphasizes precision, relevance, and data quality to enhance AI training capabilities. The dataset includes QA pairs related to factual knowledge about PFDs and PIDs, covering topics such as equipment symbols, control loops, material balances, safety interlocks, and more. Figure 2 illustrates a workflow for generating synthetic closed-ended Direct Preference Optimization (DPO) datasets. The process starts with GPT-4o generating sub-topics from a given topic, followed by creating multiple questions for each sub-topic. For every question, GPT-4o generates both chosen and rejected responses. These responses are then scored using the Nvidia-nemotron-4-340B reward model, which assigns alignment scores based on predefined dimensions. Filtering criteria are applied to retain chosen responses with scores above a threshold while discarding those below it. This structured approach ensures the generation of high-quality datasets tailored for preference optimization tasks, focusing on factual knowledge of PFDs and PIDs. Figure 3 depicts a step-by-step process for generating the SynDIP dataset, which includes descriptions of chemical process synthesis, PFDs, and PIDs for various chemicals. The workflow begins with industrial synthesis generation, where GPT-4o and Claude-3-Haiku create a process blueprint in text form, detailing raw materials, chemical reactions, operating conditions, and the equipment required for synthesis. This is followed by PFD generation, where GPT-4o and Claude-3-Haiku produce high-level process flow descriptions as text outputs. Next, PID generation uses GPT-4o and Claude-3-Haiku to develop instrumentation-level descriptions in text form. Finally, the outputs are scored and refined using the Nvidia-nemotron-4-340B reward model, ensuring the creation of high-quality synthetic datasets for chemical process engineering and industrial applications. Figure 4 illustrates a workflow for creating the LogiCore Dataset, which focuses on logical reasoning, causal and contextual comprehension, common-sense knowledge, and factual accuracy. This dataset is designed to generate relevant questions and answers based on PFDs and PIDs, improving small-scale models' ability to interpret process flows, detect errors, validate designs, and infer practical insights. Starting with a Chemical-PFD-PID QA dataset as input, models like GPT-4o and Claude-3-Haiku generate questions and corresponding answers. These outputs are evaluated using the Nvidia-nemotron-4-340B reward model, which scores them based on reasoning, knowledge, and comprehension. High-quality outputs are filtered to form the final dataset, tailored for applications requiring advanced reasoning and contextual understanding in process engineering and industrial domains. Figure 5 illustrates the Local-RAIT (Local-Retrieval Augmented Instruction-Tuning) process for generating a synthetic ChemRAIT-QA dataset, where each PDF contains detailed Chemical-PFD-PID descriptions. The workflow begins by parsing PDFs to extract text, which is divided into smaller chunks. GPT-4o generates multiple questions for each chunk, followed by detailed answers. Outputs are evaluated using the Nvidia-Nemotron-4-340B reward model for scoring. Additionally, a Large Language Model (LLM) assesses the quality of the generated question-answer pairs, potentially using a Likert scale to rate aspects such as relevance and coherence, aiding in filtering out low-quality outputs. High-quality question-answer pairs are compiled into the Synthetic ChemRAIT-QA dataset, formatted with fields for questions, context, and answers. This process ensures accurate and high-quality datasets for instruction-tuning in process engineering and chemical applications. Figure 6 illustrates the Global-RAIT workflow. Text from PDFs is parsed and divided into manageable chunks, which are grouped using hyperlink-based or semantic connections to preserve contextual relevance. GPT-4o generates questions and detailed answers for these grouped chunks, with outputs refined through a two-turn process: generating detailed long answers followed by concise short answers. A retriever selects the top-K relevant retrieval units using vector-based similarity scoring, supported by a precomputed embedding index for efficiency. Outputs are evaluated using the Nvidia-Nemotron-4-340B reward model, LLM validation, and Likert-scale feedback to ensure high-quality data. The final outputs are compiled into the Global-RAIT Synthetic Dataset, optimized for instruction-tuning tasks requiring global document comprehension and contextual accuracy.

2.6 Results Discussion

In this work, we utilize teacher-student transfer learning with large-scale language models like GPT-4o and Nvidia-nemotron-4-340B to generate and evaluate synthetic datasets for domain-specific applications, such as analyzing, interpreting, and generating PFDs. The Nvidia-nemotron-4-340B Reward model evaluates responses based on five attributes: helpfulness, correctness, coherence, complexity, and verbosity, assigning continuous scores between 0 and 4. This granularity enables effective ranking and filtering, aligning

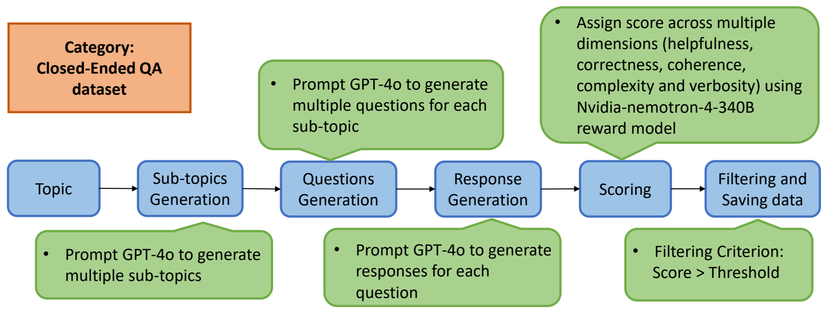

Category: Closed-Ended QA dataset
Topic
Sub-topics Generation
Questions Generation
Response Generation
Scoring
Filtering and Saving data
Prompt GPT-4o to generate multiple sub-topics
Prompt GPT-4o to generate multiple questions for each sub-topic
Prompt GPT-4o to generate responses for each question
Assign score across multiple dimensions (helpfulness, correctness, coherence, complexity and verbosity) using Nvidia-nemotron-4-340B reward model
Filtering Criterion: Score > Threshold

Figure 1: The figure shows the process for generating synthetic closed-ended QA datasets using GPT-4o and Nvidia-nemotron-4-340B reward model, including sub-topic generation, question-response creation, scoring, and filtering.

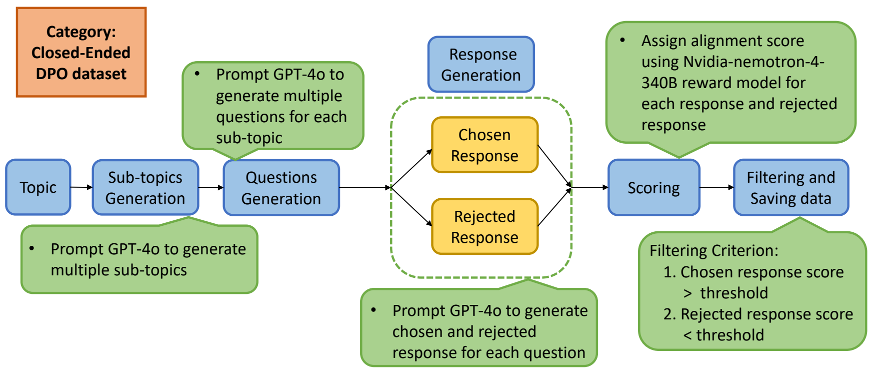

Category: Closed-Ended DPO dataset
Topic
Sub-topics Generation
Questions Generation
Response Generation
Chosen Response
Rejected Response
Scoring
Filtering and Saving data
Prompt GPT-4o to generate multiple sub-topics
Prompt GPT-4o to generate multiple questions for each sub-topic
Prompt GPT-4o to generate chosen and rejected response for each question
Assign alignment score using Nvidia-nemotron-4-340B reward model for each response and rejected response
Filtering Criterion: 1. Chosen response score > threshold 2. Rejected response score < threshold

Figure 2: The figure illustrates a workflow for generating synthetic data for closed-ended DPO datasets. GPT-4o generates sub-topics, questions, and two types of responses—"chosen" and "rejected"—for each question. Responses are scored using the Nvidia-nemotron-4-340B reward model based on coherence, correctness, and relevance. Filtering ensures quality, with chosen responses exceeding a score threshold and rejected responses falling below it. This streamlined process produces high-quality datasets optimized for alignment-sensitive tasks.

outputs with human preferences for synthetic data generation and evaluation. Figures 9 through 10 present the evaluation metrics for various datasets assessed using this model. Figure 9a shows the evaluation of closed-ended QA datasets, while Figures 9b and 9c depict scores for chosen and rejected responses in Direct Preference Optimization (DPO) datasets. Figures 9d and 9e illustrate ChemRAIT-QA and logical reasoning QA datasets, emphasizing logical consistency and contextual relevance. Figures 9f and 10a highlight Local-RAIT and Global-RAIT workflows, showcasing their ability to generate coherent question-answer pairs from contextual chunks. We perform visualizations using t-SNE and PCA to examine clustering patterns for PFDs and PIDs generated by GPT-4o, Haiku, and web data, gaining insights into semantic relationships and structural coherence. The figures7 illustrate the t-SNE and PCA visualizations of text embeddings generated by the OpenAI text-embedding-3-small model for chemical Process Flow Diagrams (PFDs) and Piping and Instrumentation Diagrams (PIDs) obtained from diverse sources: GPT-4o, Haiku, and web-retrieved data. Figures 7a and 7b present the t-SNE and PCA visualizations of embeddings for text descriptions produced by GPT-4o. The clusters in these figures are well-separated, highlighting strong semantic coherence within descriptions of similar chemical processes. PFDs and PIDs with similar operational or structural characteristics are embedded closely, while descriptions of distinct processes form

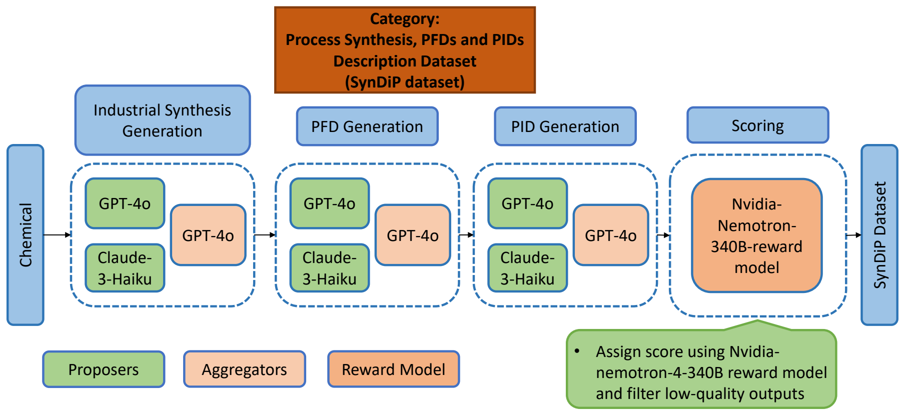

Category:
Process Synthesis, PFDs and PIDs Description Dataset (SynDiP dataset)
Industrial Synthesis Generation
PFD Generation
PID Generation
Scoring
Chemical
GPT-4o
Claude-3-Haiku
GPT-4o
Nvidia-Nemotron-340B-reward model
SynDiP Dataset
Proposers
Aggregators
Reward Model
Assign score using Nvidia-nemotron-4-340B reward model and filter low-quality outputs

Figure 3: The figure outlines the process for generating the SynDIP dataset, including process synthesis, Process Flow Diagram (PFD), and Process Instrumentation Diagram (PID) descriptions for chemicals. GPT-4o, GPT-4o-mini, and Claude-3-Haiku generate text-based outputs for synthesis, PFD, and PID descriptions, which are refined using the Nvidia-nemotron-4-340B reward model to ensure high-quality datasets.

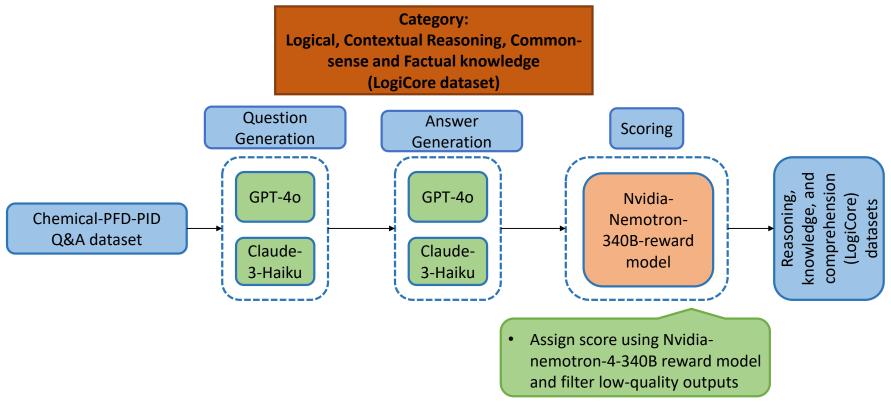

Category:
Logical, Contextual Reasoning, Common-sense and Factual knowledge (LogiCore dataset)
Question Generation
Answer Generation
Scoring
Chemical-PFD-PID Q&A dataset
GPT-4o
Claude-3-Haiku
Nvidia-Nemotron-340B-reward model
Reasoning, knowledge, and comprehension (LogiCore) datasets
Assign score using Nvidia-nemotron-4-340B reward model and filter low-quality outputs

Figure 4: The figure outlines the creation of the LogiCore Dataset, starting with a Chemical-PFD-PID QA dataset. GPT-4o and Claude-3-Haiku generate questions and answers, which are scored by the Nvidia-Nemotron-4-340B reward model. High-quality outputs are filtered to produce a dataset for advanced reasoning and contextual comprehension in process engineering..

separate clusters. Figures 7c and 7d display the t-SNE and PCA visualizations of embeddings generated by Haiku, which also show compact and distinct clusters. PFDs and PIDs related to similar chemicals or operations are grouped closely together, while unrelated processes are positioned in separate clusters. Figures 7e and 7f illustrate the t-SNE and PCA visualizations of embeddings derived from web-retrieved data. These embeddings exhibit more dispersed and overlapping clusters, reflecting less consistent semantic grouping. The overlap suggests challenges in distinguishing between PFDs and PIDs for different chemicals, potentially due to the heterogeneous and less structured nature of the web-sourced descriptions. Cluster groupings in embeddings enhance PFD and PID generation for new chemicals in few-shot prompting by leveraging similarities to identify common process components, such as unit operations (e.g., reactors, distillation columns), flow pathways (e.g., material streams, energy flows), instrumentation (e.g., control valves, pressure gauges), and equipment configurations. These shared elements enable the transfer of knowledge and provide contextually relevant examples, ensuring accuracy, efficiency, and adaptability while minimizing errors through cluster-based validation. Overall, the visualizations emphasize the grouping of semantically similar chemical PFDs and PIDs within GPT-4o and Haiku embeddings, while Web-derived embeddings show broader and noisier clustering patterns. Figure 8 highlights the similarity

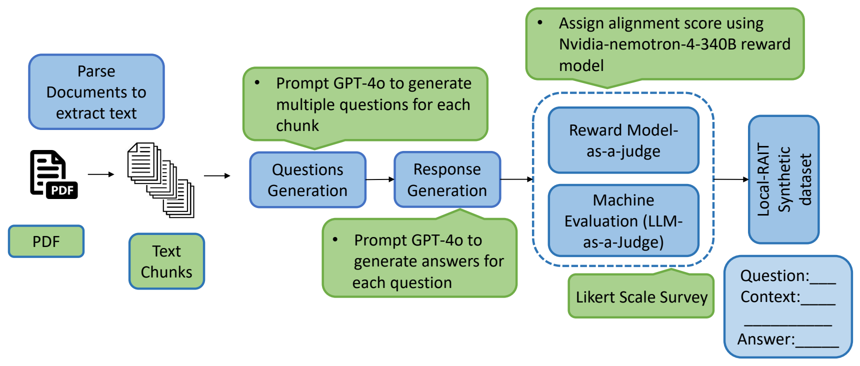

Parse Documents to extract text
PDF
PDF
Text Chunks
Prompt GPT-4o to generate multiple questions for each chunk
Questions Generation
Response Generation
Prompt GPT-4o to generate answers for each question
Assign alignment score using Nvidia-nemotron-4-340B reward model
Reward Model-as-a-judge
Machine Evaluation (LLM-as-a-Judge)
Likert Scale Survey
Local-RAIT Synthetic dataset
Question:___
Context:_____
____
Answer:_____

Figure 5: The figure shows the workflow for generating the ChemRAIT-QA Dataset: extracting text from Chemical-PFD-PID PDFs, generating question-answer pairs with GPT-4o, evaluating outputs using Nvidia-Nemotron-4-340B and LLM assessments, and compiling high-quality data for instruction-tuning in process engineering.

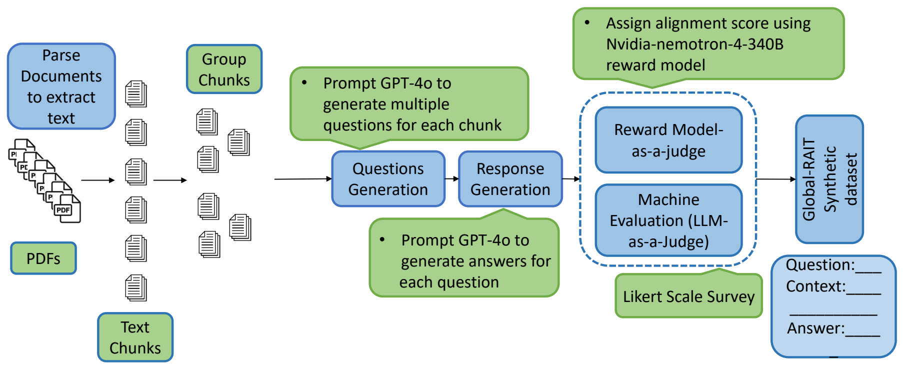

Parse Documents to extract text
PDFs
Text Chunks
Group Chunks
Prompt GPT-4o to generate multiple questions for each chunk
Questions Generation
Response Generation
Prompt GPT-4o to generate answers for each question
Assign alignment score using Nvidia-nemotron-4-340B reward model
Reward Model-as-a-judge
Machine Evaluation (LLM-as-a-Judge)
Likert Scale Survey
Global-RAIT Synthetic dataset
Question:___
Context:____
____
Answer:____

Figure 6: The figure illustrates the Global-RAIT workflow. Text from PDFs is parsed, chunked, and grouped using hyperlink-based or semantic connections to preserve context. GPT-4o generates questions and answers, refined through a two-turn process. A Retriever selects top-k relevant chunks using vector-based similarity scoring with a precomputed embedding index. Outputs are evaluated with the Nvidia-Nemotron-4-340B reward model, LLM validation, and Likert-scale feedback, creating a high-quality Global-RAIT Synthetic Dataset for instruction-tuning.

score distributions for embeddings of chemical PFD and PID descriptions across GPT-4o, Haiku, and web sources. Haiku and web embeddings show moderate alignment, with peak scores around 0.6–0.7, reflecting some consistency in process descriptions but also variability in web-derived data (Figure 8c). GPT-4o aligns more closely with web embeddings, peaking in the 0.6–0.8 range, suggesting its ability to adapt to diverse web-sourced process details (Figure 8b). GPT-4o and Haiku show the highest alignment, peaking at 0.7–0.8, demonstrating their shared capacity to represent chemical process structures with differing focus levels—Haiku on precise details and GPT-4o on broader generalization (Figure 8a).

3 Computational Time Analysis for Synthetic Dataset Generation

The computational time for generating synthetic datasets reflects the complexity of tasks required to interpret, analyze, and generate PFDs and PIDs. For all datasets, generation involves creating synthetic QA pairs, verifying outputs using the Nvidia-Nemotron-4-340B reward model, and filtering based on scores to ensure high-quality data. The Global-RAIT dataset (480.4 minutes) and Local-RAIT dataset (320.7 minutes) focus on parsing PDFs, extracting chemical process descriptions, and generating QA pairs tailored for instruction-tuning in process engineering. The DPO dataset (201.8 minutes) generates both chosen and rejected responses for factual questions, scored and filtered for preference optimization. The Closed-Ended QA dataset (155.4 minutes) generates factual QA pairs on topics like material balances and equipment symbols, making it the most efficient. LogiCore (600.6 minutes) emphasizes logical reasoning and contextual comprehension by generating QA pairs to detect errors, validate designs, and infer insights from PFDs and PIDs. Knowledge-Graph (551.3 minutes) supports structured chemical process knowledge integration and validation. SynDiP (2179.6 minutes) requires the most time due to its detailed workflows for generating chemical process synthesis descriptions, PFDs, and PIDs, with all outputs verified and refined to ensure precision and quality. This distribution highlights the trade-offs between dataset complexity, computational time, and the depth of information tailored for fine-tuning small-scale models (Figure 11a).

4 Carbon Emissions for Dataset Generation

Carbon emissions for dataset generation vary significantly across tasks, reflecting differences in computational complexity and resource requirements. SynDiP has the highest emissions, at 1.25 kg CO_2, due to its resource-intensive process of generating detailed chemical process descriptions. Moderate emissions are observed for LogiCore (0.34 kg CO_2), Global-RAIT (0.26 kg CO_2), and Knowledge-Graph (0.27 kg CO_2), which involve complex reasoning, retrieval, and graph structuring. In contrast, Closed-Ended QA (0.18 kg CO_2), DPO (0.22 kg CO_2), and Local-RAIT (0.18 kg CO_2) exhibit the lowest emissions, highlighting their efficiency in handling simpler or more targeted dataset generation tasks. To measure these emissions, tools such as *CodeCarbon* are employed. *CodeCarbon* tracks real-time energy consumption and estimates CO_2 emissions based on hardware usage (e.g., CPU, GPU) and the carbon intensity of the electricity grid. This enables developers to quantify the environmental impact of dataset generation and optimize workflows for greater energy efficiency. Reporting emissions in kg CO_2 provides transparency and supports efforts to balance computational demands with sustainability goals (Figure 11b).

5 Performance Analysis of Agentic Conversational Framework

Our study evaluates the effectiveness of an agentic conversational framework for generating structured and domain-specific responses, particularly in chemical process engineering. The experiments span multiple dimensions, including fine-tuning, retrieval-augmented generation (Graph RAG), feedback mechanisms, and generalization to unseen chemical data. The performance assessment is based on a 1500-question test dataset, with models evaluated using Nvidia-Nemotron-4-340B reward model scores and various linguistic metrics such as BLEU, METEOR, ROUGE, BERTScore, SacreBLEU, and overall similarity.

5.1 Impact of Fine-Tuning and Graph RAG

The initial set of evaluations compared six framework configurations incorporating fine-tuning, Graph RAG, and their absence to quantify their influence on helpfulness, correctness, coherence, complexity, and verbosity. The results indicate that fine-tuning significantly enhances performance across all five quality metrics, enabling the model to generate more accurate, structured, and contextually rich responses. Graph RAG further improves correctness and coherence, particularly in non-fine-tuned models, by retrieving relevant knowledge to support response generation. Models that lacked both fine-tuning and Graph RAG exhibited the lowest performance, highlighting their dependence on intrinsic pretraining knowledge, which is insufficient for specialized domains like chemical process synthesis.

5.2 Comparison with GPT-4o

To benchmark the agentic framework's performance, we compared a fine-tuned Llama 1B model acting as a meta-agent against GPT-4o, the gold-standard LLM. GPT-4o consistently outperformed the agentic framework across all evaluation metrics, particularly in helpfulness and correctness, which reflects its superior pretrained knowledge and reasoning capabilities. However, the fine-tuned Llama 1B model demonstrated competitive performance, particularly in coherence and complexity, indicating that task-specific fine-tuning enables the agentic framework to generate structured and logically connected responses even without the vast knowledge base of GPT-4o.

5.3 Pretraining Variations: Influence of Graph RAG and Feedback

Further evaluations assessed pretrained Llama models under three different setups: (i) with Graph RAG and feedback, (ii) without Graph RAG but with feedback, and (iii) without both Graph RAG and feedback. The results reinforce that Graph RAG improves retrieval-based augmentation, ensuring that the generated responses maintain factual accuracy, while feedback mechanisms refine response structure over time. Models without Graph RAG and feedback performed the worst, particularly in correctness and coherence, demonstrating that pretrained models alone lack the ability to adapt to domain-specific constraints without retrieval augmentation or iterative optimization.

5.4 Evaluation of Linguistic and Similarity Metrics

A detailed analysis using linguistic and similarity metrics such as BERTScore, BLEU, METEOR, ROUGE (1, 2, L), SacreBLEU, and overall similarity provides further insights into the framework's textual and semantic alignment with reference answers. Fine-tuned models consistently outperformed non-fine-tuned configurations across all these metrics, confirming that fine-tuning plays a dominant role in improving lexical precision, phrase structure, and semantic similarity. Graph RAG moderately enhances performance, particularly in BERTScore and ROUGE metrics, suggesting that retrieval augmentation aids in improving factual recall but does not fully compensate for fine-tuning. Smaller models like SmolLM 135M scored significantly lower across all linguistic metrics, indicating their limitations in handling complex, structured queries.

5.5 Generalization to Unseen Chemical Data

To test the framework's ability to handle previously unseen data, we evaluated its performance on chemical compounds not included in fine-tuning. The fine-tuned Llama 1B model with Graph RAG and feedback achieved the highest scores, demonstrating strong generalization capabilities, whereas non-fine-tuned models struggled to adapt, particularly in correctness and coherence. Comparisons with GPT-4o confirm its superior generalization abilities, but the fine-tuned Llama 1B model remains competitive in complexity and verbosity, indicating that task-specific optimization can bridge the gap between proprietary and open-source LLMs.

Additional evaluations assessed how different framework configurations (fine-tuning, Graph RAG, and feedback) influenced generalization. Results show that feedback plays a crucial role in refining responses for unseen chemicals, while Graph RAG helps maintain factual relevance even in non-fine-tuned models. The lowest-performing models were those without fine-tuning, Graph RAG, or feedback, highlighting that pretrained models alone struggle with novel domain-specific data.

6 Impact of Width and Depth Pruning on Model Performance and Efficiency

To analyze the trade-off between model efficiency and response quality, we conducted width pruning and depth pruning experiments on our agentic conversational framework. These pruning techniques were applied at varying percentages to evaluate their effects on response quality, computational time, and generalization to unseen chemical data. The first set of experiments assessed how width and depth pruning influence model performance on the test dataset. Width pruning, which reduces the number of neurons per layer, leads to a gradual decline in correctness and complexity, with more aggressive pruning causing significant performance degradation. Depth pruning, which removes entire layers from the model, affects coherence and correctness more severely, as fewer layers limit the model's ability to retain hierarchical information and maintain logical flow. Higher pruning percentages in both methods lead to substantial drops in response quality, though width pruning preserves coherence better than depth pruning. To further examine the framework's robustness, we evaluated width and depth-pruned models on an unseen chemical dataset. Similar trends were observed, with width pruning leading to noticeable reductions in correctness and complexity, while depth pruning had a stronger negative impact on coherence. The results suggest that pruning reduces the framework's ability to generalize to new data, with larger pruning percentages causing significant deterioration across all evaluation metrics. However, moderate levels of width pruning retained better performance compared to depth pruning, indicating that reducing neuron count within layers is less destructive than removing entire layers. In addition to response quality, we measured the computational time savings achieved through pruning. Both width and depth pruning significantly reduced processing time, with higher pruning levels leading to more pronounced efficiency gains. Width pruning showed a more linear reduction in computational time, while depth pruning resulted in larger time savings at higher pruning levels. These results confirm that pruning is an effective strategy for improving computational efficiency, though it comes at the cost of reduced response accuracy and generalization capability.

7 Experimental Settings

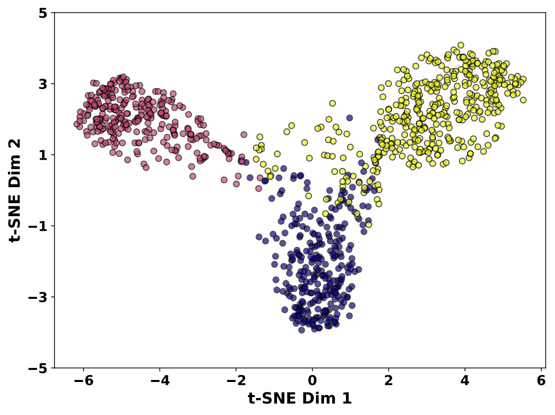
The fine-tuning process for the Meta LLaMA 3.2-1B model was carried out in three sequential stages, progressively refining the model's ability to generate responses by leveraging different datasets. The first phase involved fine-tuning on a QA dataset, where the model learned fundamental question-answering abilities. This was followed by Direct Preference Optimization (DPO) fine-tuning, which improved response alignment by distinguishing between preferred and rejected responses. Finally, the model was further enhanced through Retrieval-Augmented Instruction Tuning (RAIT), which integrated retrieved contextual information to support multi-hop reasoning and knowledge grounding.

For all three stages, training was conducted using Low-Rank Adaptation (LoRA), which enabled efficient parameter tuning while keeping the majority of model weights frozen. LoRA was configured with a rank of 16, a scaling factor of 16, and no dropout, optimizing both performance and memory efficiency. The model was trained with a batch size of 2 per device, using 4 gradient accumulation steps to achieve an effective batch size of 8. The AdamW optimizer with 8-bit quantization was employed, using a learning rate of 2×10^{-4} with a linear decay schedule and a weight decay of 0.01. Mixed precision training was applied with FP16 or BF16, depending on hardware compatibility, and all experiments were executed on NVIDIA Tesla T4 GPUs. To handle long-context inputs effectively, all fine-tuning steps were conducted with a maximum sequence length of 4096 tokens.

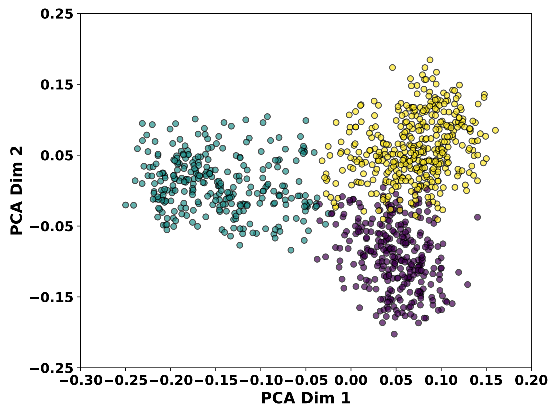
Initially, QA and RAIT fine-tuning were performed for 5 epochs, while DPO was trained for 2 epochs. However, the model did not reach convergence in these initial runs. To improve performance, additional fine-tuning was conducted, extending QA and RAIT training to 15 epochs and DPO to 5 epochs. This additional training allowed the model to better capture instruction-following patterns, refine preference-based response ranking, and integrate contextual information from retrieved knowledge.

The entire fine-tuning pipeline was implemented in Python using PyTorch and Unsloth, leveraging gradient checkpointing to reduce memory consumption. Datasets were processed with the Hugging Face `datasets` library, and model training was managed using `SFTTrainer` for supervised fine-tuning and `DPOTrainer` for preference optimization. Performance was evaluated across all stages using standard NLP metrics, including BLEU, ROUGE, METEOR, BERTScore, and Similarity metric.

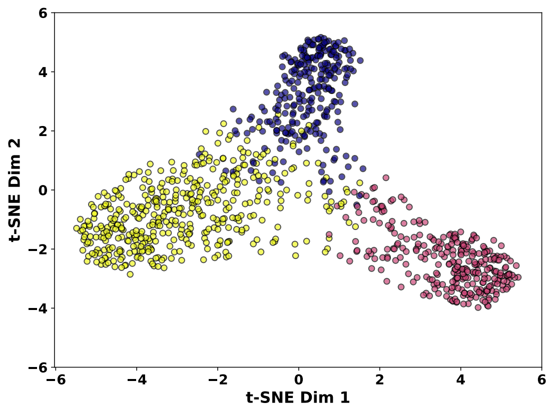
Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009



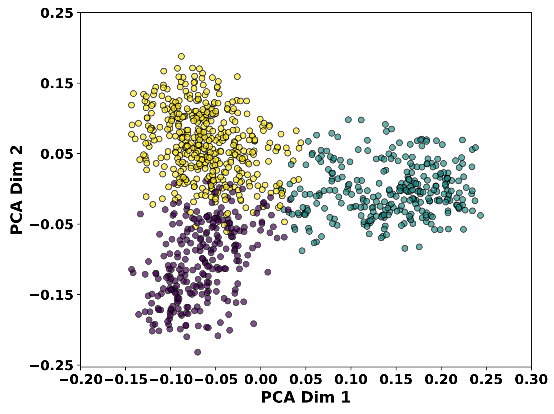
(a) t-SNE visualization of embeddings generated by the OpenAI text-embedding-3-small model for text descriptions produced by GPT-4o. Clusters indicate semantically similar groupings of process flow and instrumentation diagram descriptions.



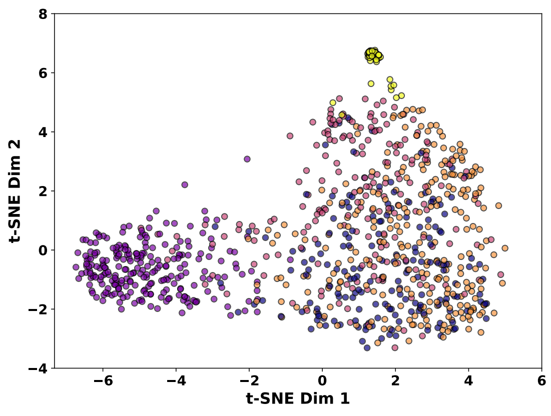
(b) Principal Component Analysis (PCA) plot for embeddings of text descriptions produced by GPT-4o. The first two principal components capture the largest variance, revealing patterns in the embedding space.



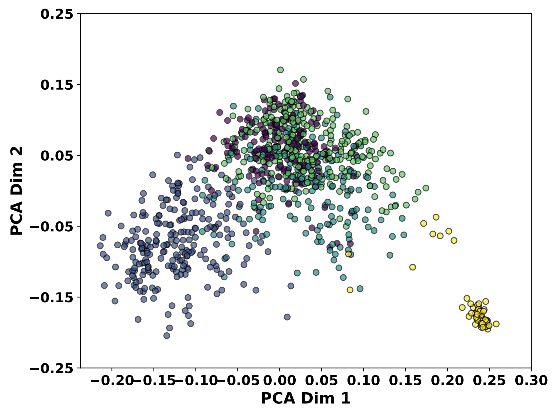
(c) t-SNE visualization of text embeddings generated by Haiku for chemical PFDs and PIDs. Distinct clusters represent semantic similarities in the embedding space.



(d) PCA visualization of text embeddings generated by Haiku, showing variance patterns across the first two principal components.

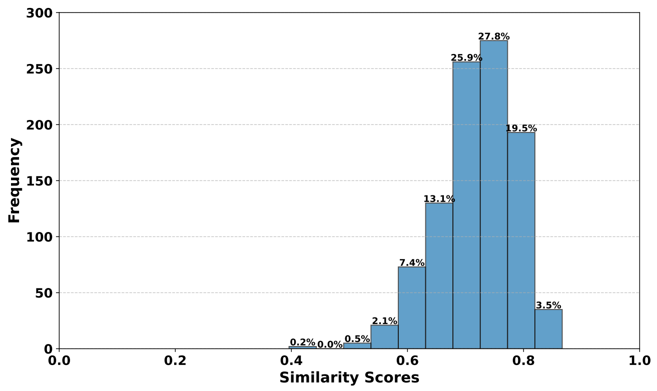


(e) t-SNE visualization of text embeddings for chemical PFDs and PIDs derived from web-retrieved data. Clusters show grouping in two-dimensional space.

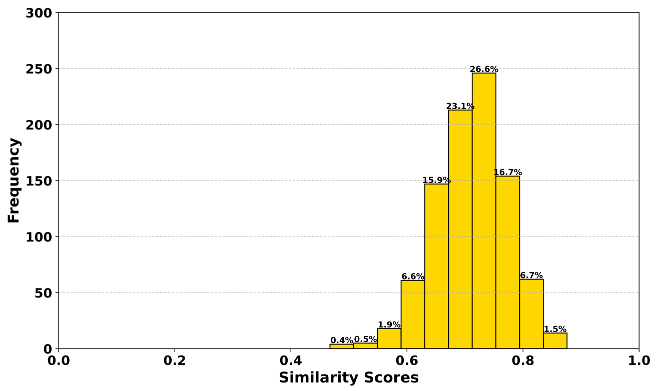


(f) PCA visualization of text embeddings for chemical PFDs and PIDs derived from web data, highlighting main variance directions in embeddings.

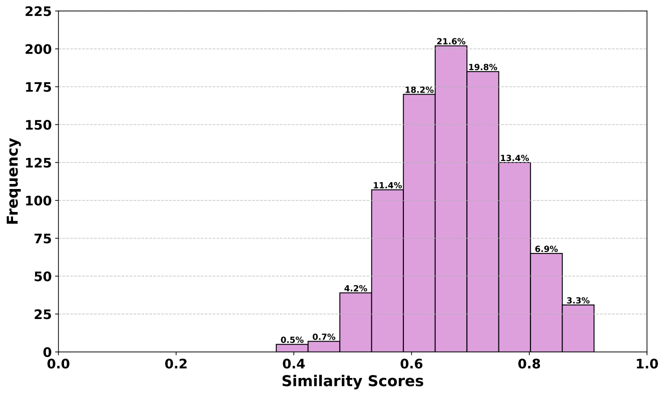
Figure 7: t-SNE and PCA visualizations of text embeddings for PFDs and PIDs, showing clustering patterns for GPT-4o, Haiku, and web-retrieved data.



(a) Distribution of similarity scores between embeddings generated by GPT-4o and Haiku. The peak at 0.7–0.8 indicates strong semantic alignment, highlighting closer coherence compared to web-retrieved data embeddings

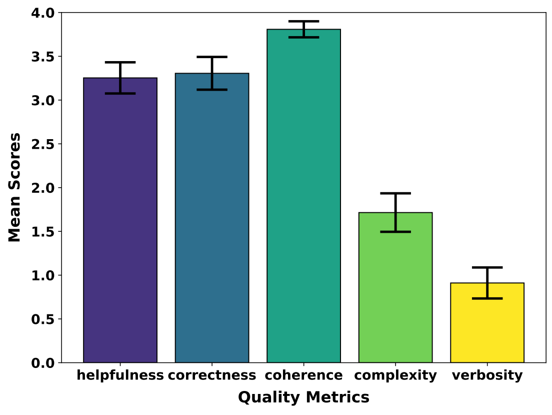


(b) Distribution of similarity scores between embeddings generated by GPT-4o and web-retrieved data. The peak at 0.6–0.7 reflects moderate semantic alignment, with lower and higher similarity scores being less frequent

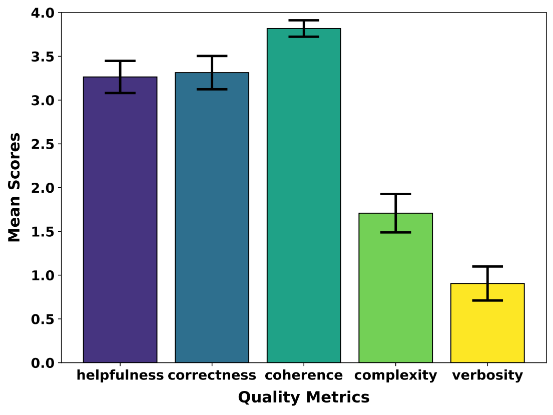


(c) Distribution of similarity scores between embeddings generated by Haiku and web-retrieved data. Moderate alignment is observed, with a peak at 0.6–0.7 and broader dispersion compared to GPT-4o embeddings.

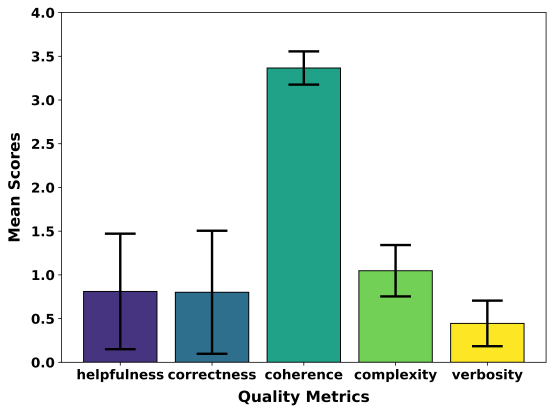
Figure 8: Histogram similarity plots showing embeddings of chemical PFDs and PIDs, highlighting semantic alignment across GPT-4o, Haiku, and web data.



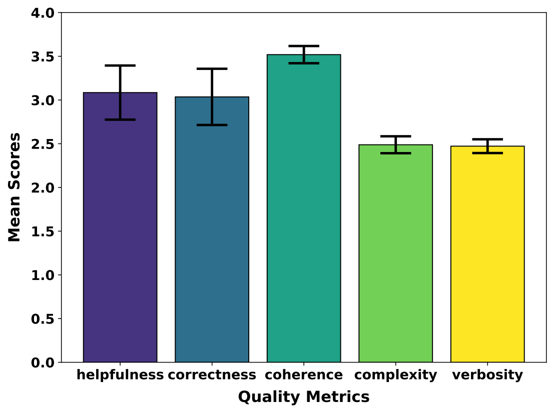
(a) Quality metrics for closed-ended QA dataset generation, evaluated using the Nvidia-nemotron-4-340B Reward model across dimensions of helpfulness, correctness, coherence, complexity, and verbosity



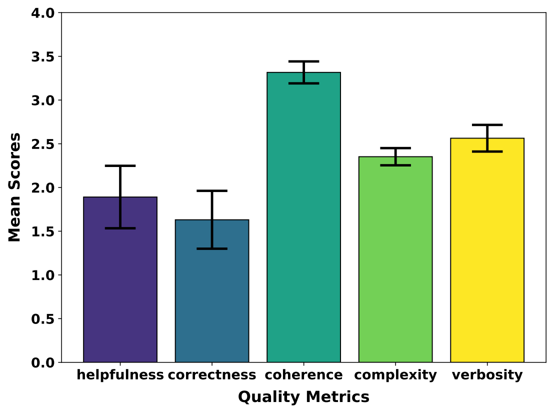
(b) Quality metrics for chosen responses in the Direct Preference Optimization (DPO) dataset generation, scored using the Nvidia-nemotron-4-340B Reward model to ensure high-quality outputs



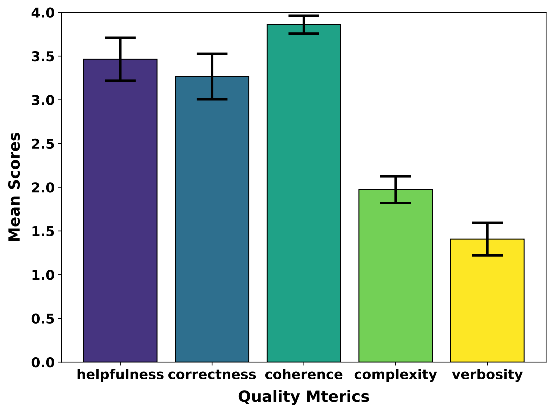
(c) Quality metrics for rejected responses in the Direct Preference Optimization (DPO) dataset generation, highlighting evaluation by the Nvidia-nemotron-4-340B Reward model for filtering low-quality responses



(d) Evaluation of ChemRAIT-QA dataset generation, showing quality metrics scored using the Nvidia-nemotron-4-340B Reward model to ensure relevance and coherence

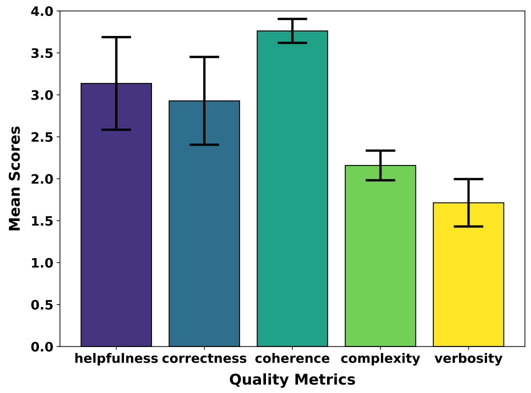


(e) Quality metrics for logical reasoning QA dataset generation, scored by the Nvidia-nemotron-4-340B Reward model across dimensions of logical consistency, causal comprehension, and contextual relevance



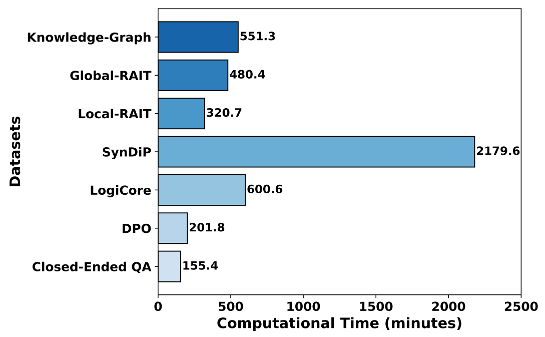
(f) Evaluation metrics for the Local-RAIT synthetic dataset, highlighting mean scores for helpfulness, correctness, coherence, complexity, and verbosity as assessed by the Nvidia-nemotron-4-340B Reward model

Figure 9: Evaluation metrics for various synthetic datasets assessed using the Nvidia-nemotron-4-340B Reward model across multiple quality dimensions.

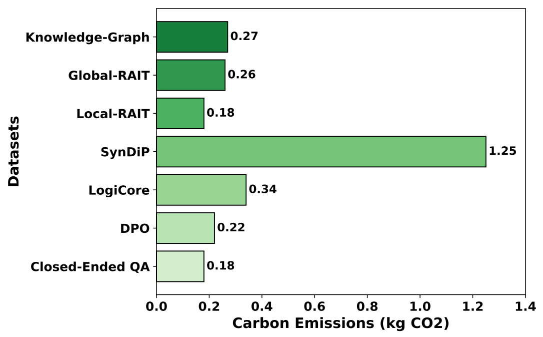


(a) Evaluation of the Global-RAIT synthetic dataset generation, showing metrics for helpfulness, correctness, coherence, complexity, and verbosity as scored by the Nvidia-nemotron-4-340B Reward model

Figure 10: Evaluation metrics for Global-RAIT synthetic dataset using the Nvidia-nemotron-4-340B Reward model.

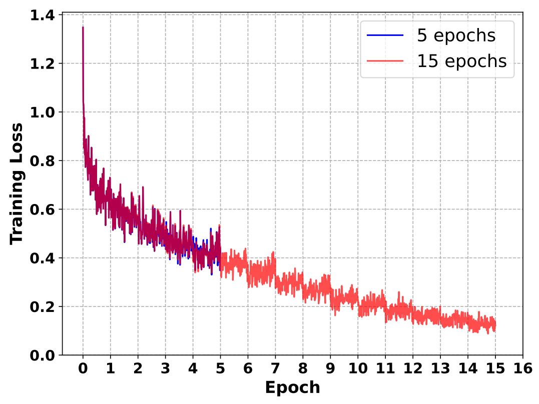


(a) Computational time required to generate synthetic datasets for PFDs and PIDs, involving QA pair creation, verification using the Nvidia-Nemotron-4-340B reward model, and quality filtering. SynDiP requires the most time due to its complex workflows, while Closed-Ended QA is the fastest to generate.

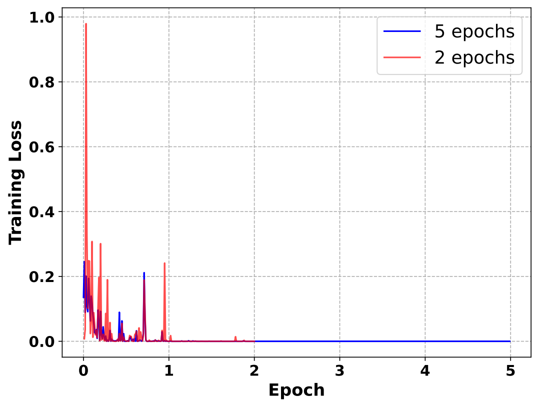


(b) Carbon emissions (kg CO_2) for dataset generation, with SynDiP showing the highest emissions and Closed-Ended QA, DPO, and Local-RAIT the lowest

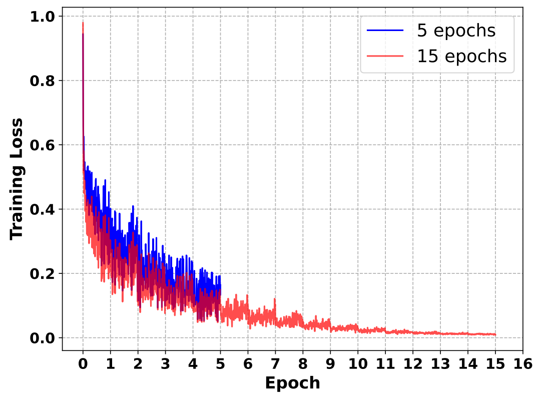
Figure 11: Analysis of synthetic dataset generation: (a) computational time required for generating datasets, and (b) associated carbon emissions.



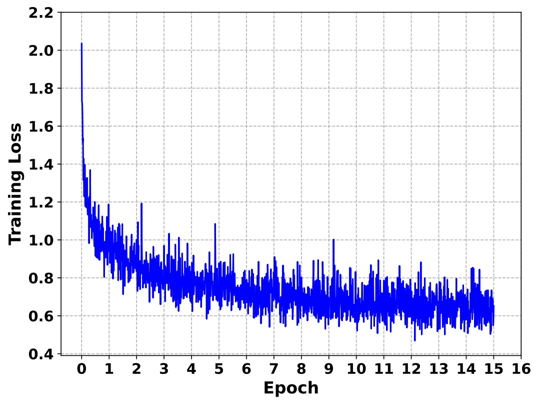
(a) Training loss curves for the Llama 3.2 1B model fine-tuned using SynDIP (chemical synthesis) and LogiCore (reasoning and comprehension) datasets. The 15-epoch configuration achieves better convergence below 0.2, while the 5-epoch configuration stabilizes at a higher loss.



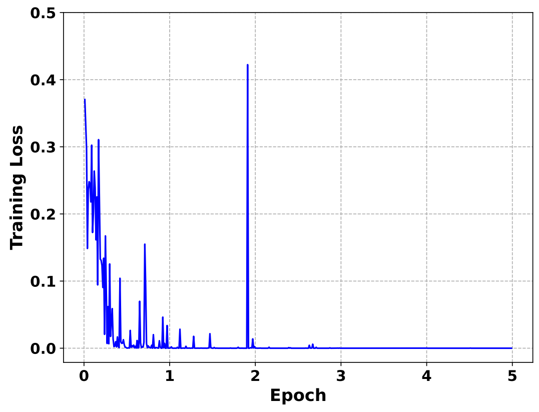
(b) Training loss curves for the Llama 3.2 1B model fine-tuned on DPO tasks. The loss decreases rapidly from 0.8 to near-zero within 2 epochs, with extended training to 5 epochs maintaining near-zero loss.



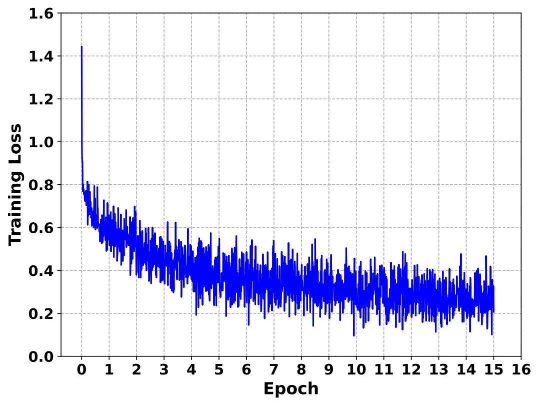
(c) Training loss curves for the Llama 3.2 1B model fine-tuned on RAG tasks using Local-RAIT and Global-RAIT datasets. The model achieves near-zero convergence within 15 epochs.



(d) Training loss curves for the Llama 3.2 1B model fine-tuned using SynDIP (chemical synthesis) and LogiCore (reasoning and comprehension) datasets. The 15-epoch configuration achieves better convergence below 0.2, while the 5-epoch configuration stabilizes at a higher loss.

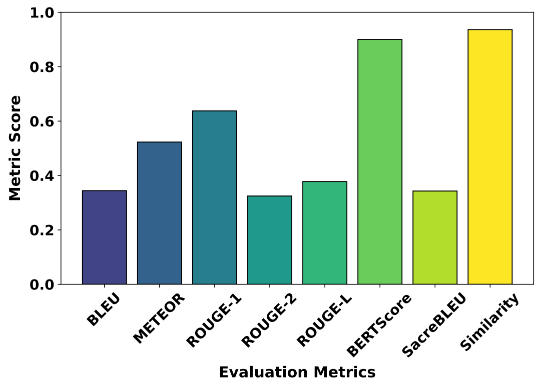


(e) Training loss curves for the Llama 3.2 1B model fine-tuned on DPO tasks. The loss decreases rapidly from 0.8 to near-zero within 2 epochs, with extended training to 5 epochs maintaining near-zero loss.

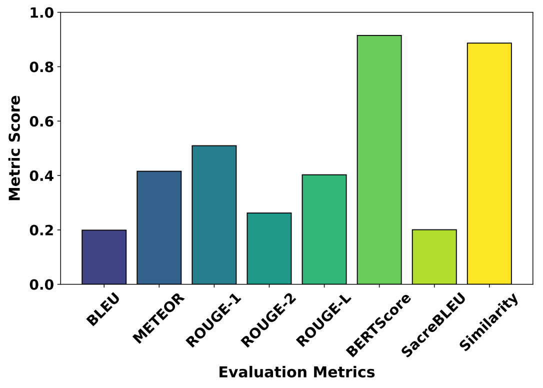


(f) Training loss curves for the Llama 3.2 1B model fine-tuned on RAG tasks using Local-RAIT and Global-RAIT datasets. The model achieves near-zero convergence within 15 epochs.

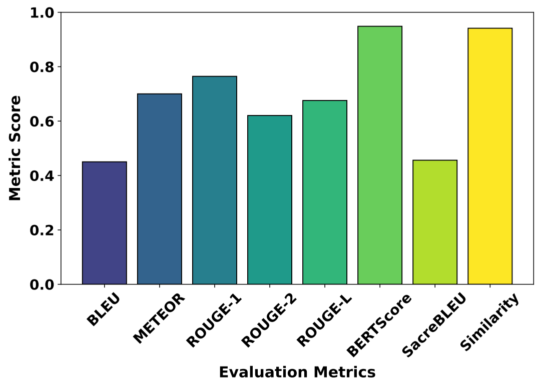
Figure 12: The figure shows training loss curves for Llama 3.2 1B and SmolLM2-135M-Instruct models, fine-tuned with QLoRA on various datasets. The curves highlight the impact of extended training and memory-efficient techniques.



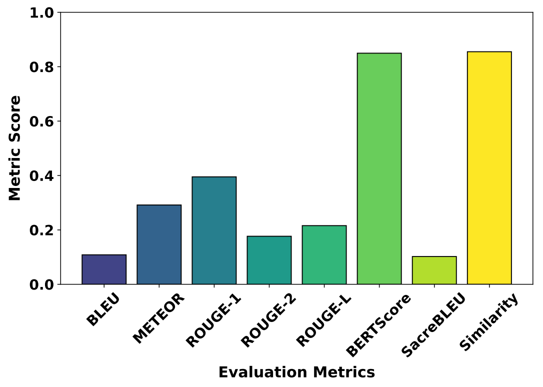
(a) Performance metrics of the fine-tuned Llama 3.2 1B model on QA tasks using evaluation sets from ChemRAIT-QA and LogiCore datasets. Metrics such as BLEU, ROUGE, BERTScore, and Similarity validate the robustness of fine-tuning for domain-specific QA tasks.



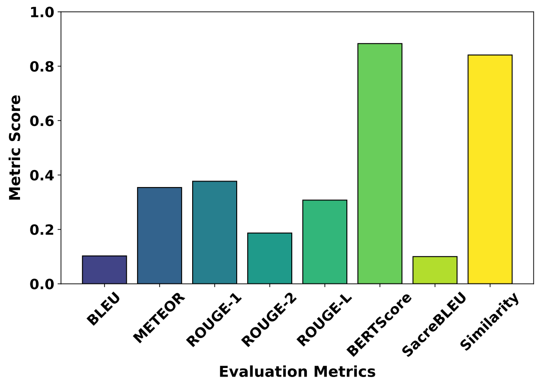
(b) Evaluation metrics for the fine-tuned Llama 3.2 1B model on DPO tasks. Metrics such as BLEU, ROUGE, BERTScore, Meteor, and SacreBLEU demonstrate the success of preference-based fine-tuning in aligning outputs with correctness, coherence, and relevance.



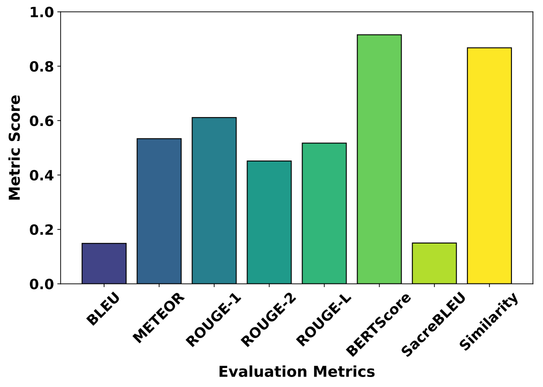
(c) Evaluation metrics for the fine-tuned Llama 3.2 1B model on RAG tasks using Local-RAIT and Global-RAIT datasets. Metrics such as BLEU, ROUGE, BERTScore, and Similarity show effective retrieval-augmented workflows for precise and relevant outputs.



(d) QA task performance of SmolLM2-135M-Instruct on ChemRAIT-QA and LogiCore datasets. High BLEU, ROUGE, BERTScore, and Similarity scores highlight the model's contextual accuracy and coherence.

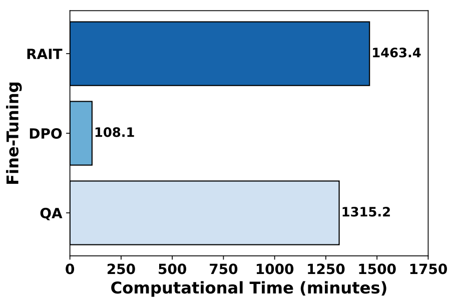


(e) Evaluation metrics for SmolLM2-135M-Instruct on DPO tasks. BLEU, ROUGE, BERTScore, Meteor, and SacreBLEU confirm superior alignment and factual correctness.

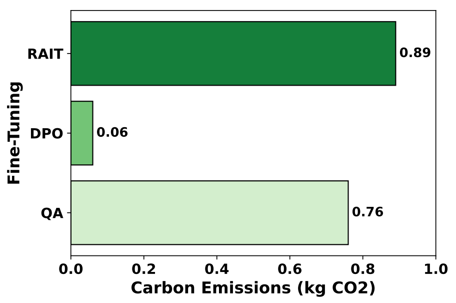


(f) RAG task performance of SmolLM2-135M-Instruct using Local-RAIT and Global-RAIT datasets. High BLEU, ROUGE, and BERTScore metrics confirm the model's retrieval-augmented response generation effectiveness.

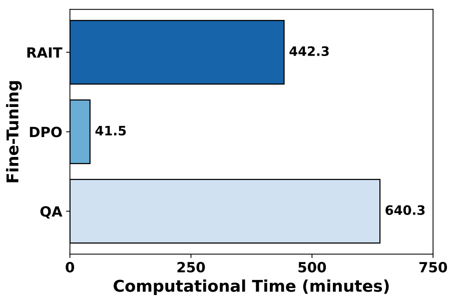
Figure 13: The figure shows performance evaluation metrics for the fine-tuned Llama 3.2 1B and SmolLM2-135M-Instruct models on QA, DPO, and RAG tasks. The metrics validate fine-tuning effectiveness across various tasks.



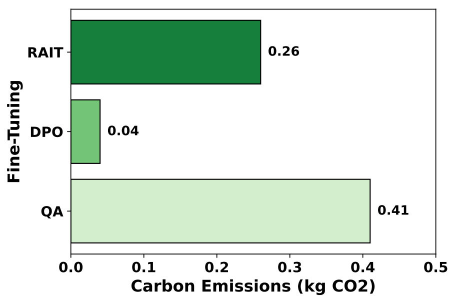
(a) Computational time required to fine-tune the Llama 3.2 1B model on QA, DPO, and RAIT tasks. RAIT takes the longest, followed by QA, with DPO being the fastest.



(b) Carbon emissions generated during fine-tuning of the Llama 3.2 1B model for QA, DPO, and RAIT tasks. RAIT shows the highest emissions, followed by QA, with DPO being the lowest.

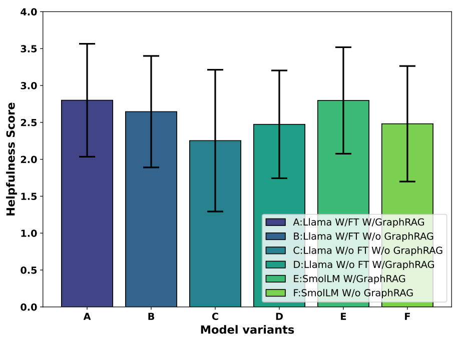


(c) Computational time required to fine-tune the SmolLM2-135M-Instruct model on QA, DPO, and RAIT tasks. RAIT takes the longest, followed by QA, with DPO being the fastest.

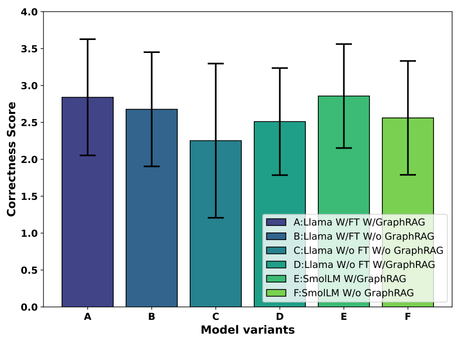


(d) Carbon emissions generated during fine-tuning of the SmolLM2-135M-Instruct model for QA, DPO, and RAIT tasks. RAIT has the highest emissions, followed by QA, with DPO being the lowest.

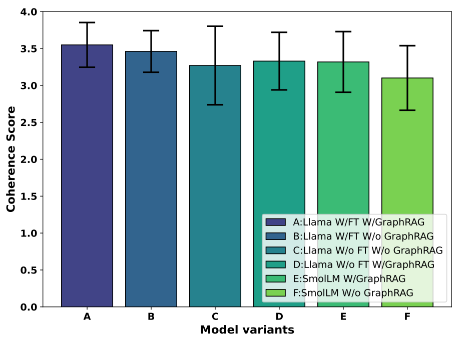
Figure 14: Fine-tuning time and carbon emissions for Llama 3.2 1B and SmolLM2-135M-Instruct models on QA, DPO, and RAIT tasks. The metrics highlight the resource efficiency of each model across tasks.



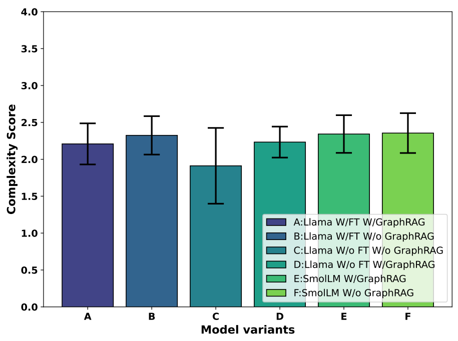
(a) Helpfulness scores of framework variants on the test dataset. Fine-tuning and Graph RAG achieve higher helpfulness scores, with Llama 1B variants slightly outperform SmolLM 135M.



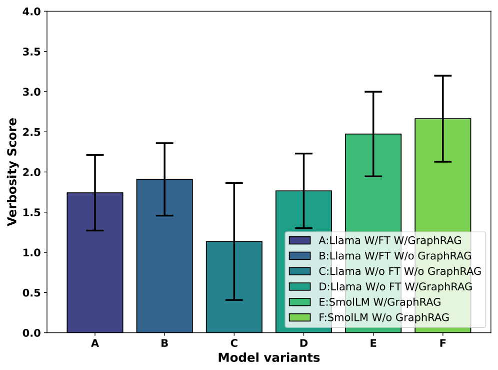
(b) Correctness scores of framework variants on the test dataset.



(c) Coherence scores of framework variants on the test dataset. Fine-tuning and Graph RAG improve coherence, with Llama 1B models producing more structured and logically connected responses.



(d) Complexity scores of framework variants on the test dataset. Fine-tuning increases response complexity, while Graph RAG helps maintain structured and informative content.



(e) Verbosity scores of framework variants on the test dataset. Fine-tuning increases verbosity, while Graph RAG helps regulate verbosity levels for more concise responses.

Figure 15: Performance comparison of framework variants on the test dataset. The figures illustrate the evaluation of six framework variants on a 1500-question dataset, assessed using the Nvidia-Nemotron-4-340B reward model across five key metrics: helpfulness, correctness, coherence, complexity, and verbosity. The results demonstrate the impact of fine-tuning and Graph RAG, with Llama 1B and SmolLM 135M across all metrics.

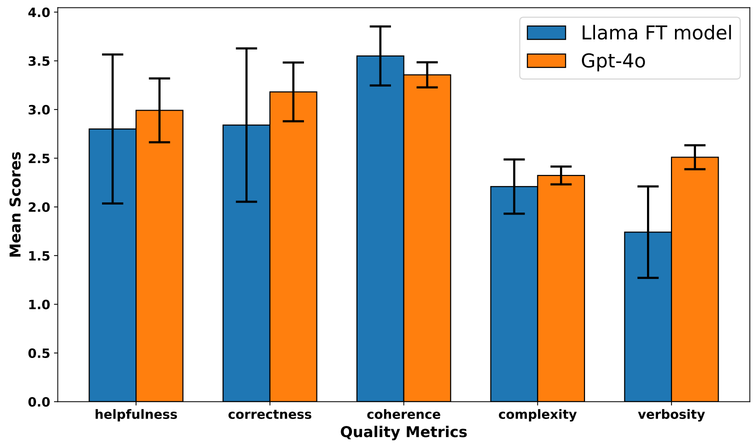


Figure 16: Comparison of Llama 1B-based agentic conversational framework with GPT-4o. The figure illustrates the performance of the fine-tuned Llama 1B model as a meta-agent against the GPT-4o gold-standard LLM across five quality metrics: helpfulness, correctness, coherence, complexity, and verbosity.

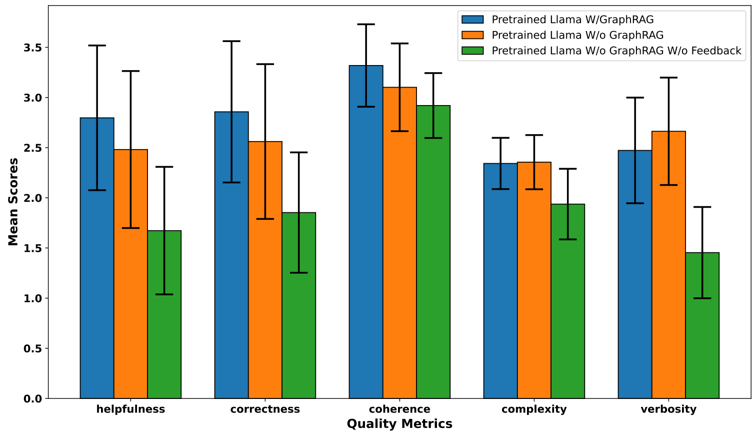
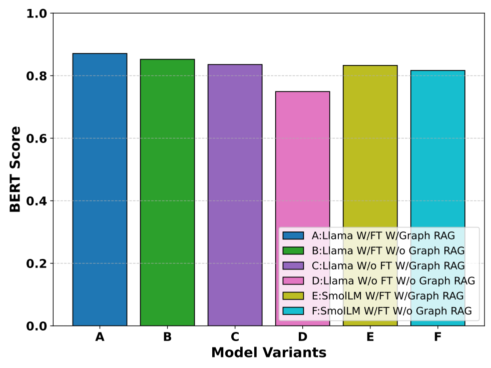
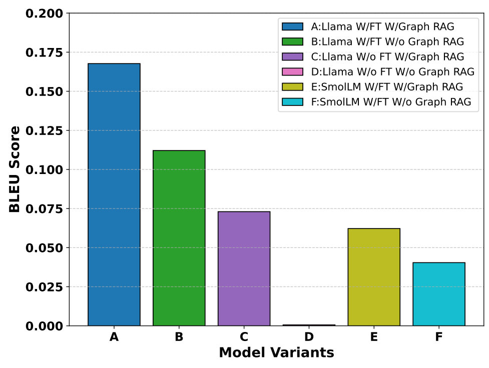


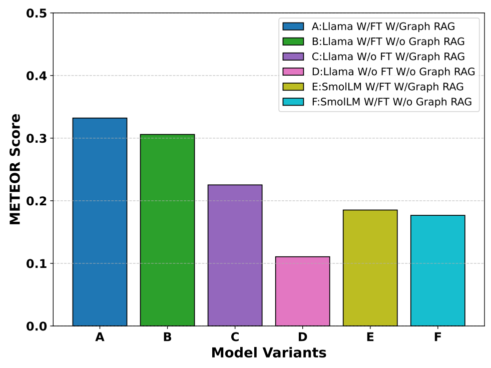
Figure 17: Performance comparison of pretrained Llama under different setups. The figure evaluates pretrained Llama models with Graph RAG and feedback, without Graph RAG but with feedback, and without both Graph RAG and feedback, across five quality metrics. Graph RAG and feedback together yield the best performance, while removing both significantly degrades response quality.



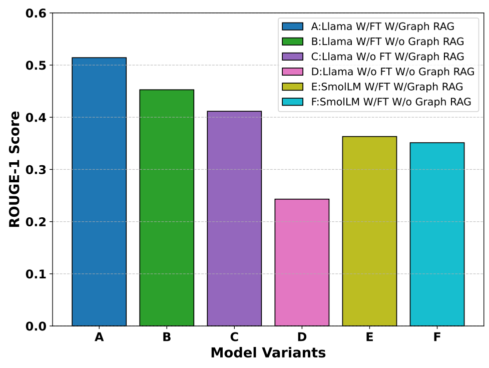
(a) BERT scores of framework variants on the test dataset. Fine-tuning and Graph RAG improve semantic similarity, with Llama 1B achieving the highest alignment with reference answers



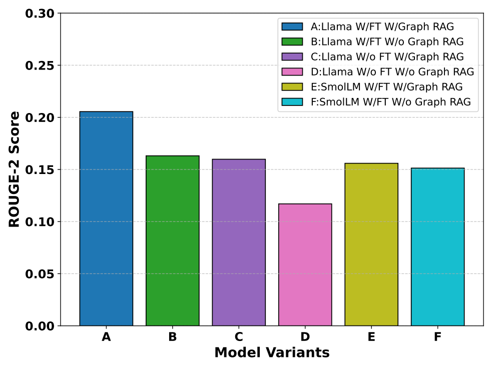
(b) BLEU scores of framework variants on the test dataset. Fine-tuning improves lexical overlap with reference answers, with Llama 1B models achieving higher BLEU scores than SmolLM 135M.



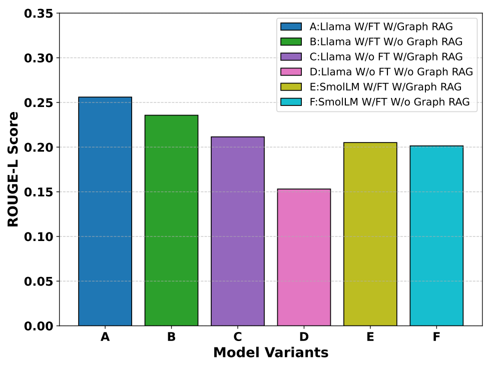
(c) METEOR scores of framework variants on the test dataset. Fine-tuning and Graph RAG improve semantic and morphological alignment, with Llama 1B models achieving the highest scores.



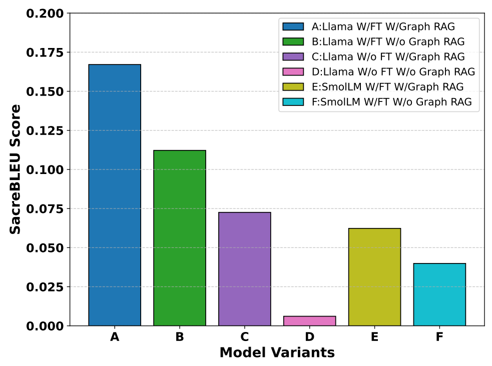
(d) ROUGE-1 scores of framework variants on the test dataset. Fine-tuning improves unigram recall, with Llama 1B models achieving better content overlap with reference answers.



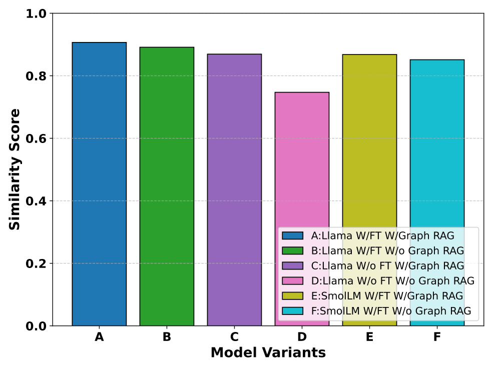
(e) ROUGE-2 scores of framework variants on the test dataset. Fine-tuning enhances bigram recall, with Llama 1B models achieving higher phrase-level similarity to reference answers.



(f) ROUGE-L scores of framework variants on the test dataset. Fine-tuning improves longest common subsequence recall, with Llama 1B models achieving better structural alignment with reference answers.

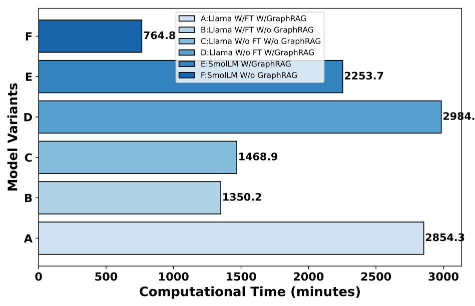


(a) SacreBLEU scores of framework variants on the test dataset. Fine-tuning improves translation-style similarity, with Llama 1B models achieving the highest alignment with reference answers.

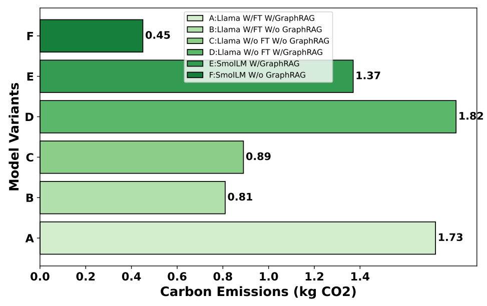


(b) Similarity scores of framework variants on the test dataset. Fine-tuning and Graph RAG improve response similarity to reference answers, with Llama 1B models achieving the highest alignment.

Figure 18: Evaluation metrics for framework variants on the test dataset. The figures present a comparative analysis of six framework configurations using multiple evaluation metrics, including BERT, BLEU, METEOR, ROUGE (1, 2, L), SacreBLEU, and Similarity scores. Fine-tuning consistently improves performance across all metrics, while Graph RAG enhances contextual accuracy and retrieval-based augmentation. Llama 1B models outperform SmolLM 135M across all configurations, with the best results achieved using both fine-tuning and Graph RAG.

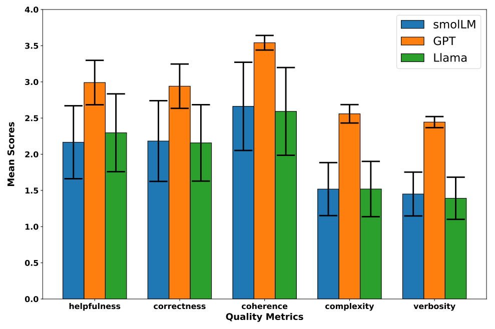


(a) Testing time for the agentic framework on the test dataset. Fine-tuned Llama models take the longest, while SmolLM models complete testing significantly faster.

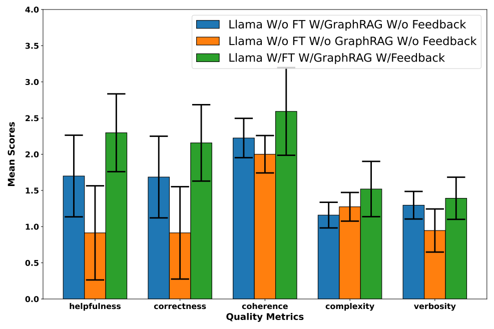


(b) Carbon emissions from testing the agentic framework on the test dataset. Fine-tuned Llama models with Graph RAG generate the highest emissions, while SmolLM models result in the lowest environmental impact.

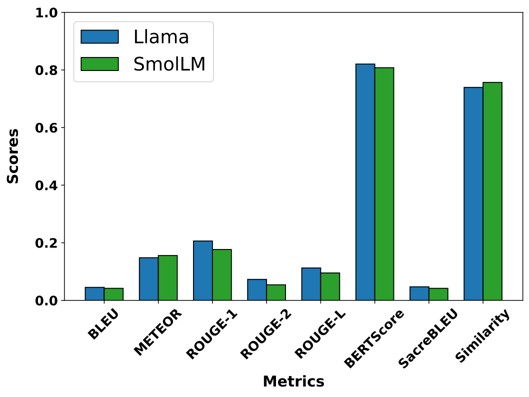
Figure 19: Computational cost and carbon footprint of testing the agentic framework. The figures illustrate the time required and carbon emissions for processing the test dataset across different framework variants. Fine-tuned Llama models with Graph RAG exhibit the highest computational cost and emissions, while SmolLM models are the most efficient



(a) Performance of framework variants on previously unseen chemicals. The figure compares SmolLM, Llama 1B, and GPT-4o across five quality metrics, evaluating their ability to generalize to unseen chemical data. Llama 1B achieves strong performance, but GPT-4o leads in overall response quality.

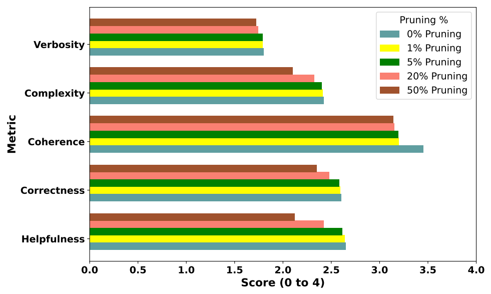


(b) Evaluation of framework performance on unseen chemicals using linguistic and similarity metrics. The figure compares Llama 1B and SmolLM across multiple evaluation metrics, with Llama 1B achieving significantly higher scores, indicating better generalization to unknown chemical data.

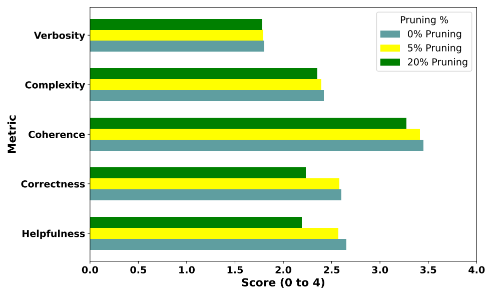


(c) Impact of fine-tuning, Graph RAG, and feedback on framework performance for unseen chemicals. The figure compares Llama 1B with and without fine-tuning, Graph RAG, and feedback, demonstrating that fine-tuning combined with Graph RAG and feedback significantly improves response quality for unknown chemicals.

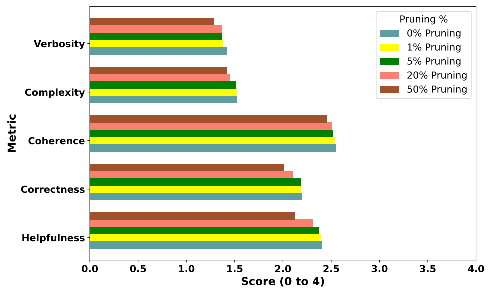
Figure 20: Generalization performance of framework variants on unseen chemical data. The figures compare different framework configurations on a test dataset of previously unseen chemicals, evaluating response quality across multiple metrics. Fine-tuning, Graph RAG, and feedback significantly improve adaptability, with Llama 1B outperforming SmolLM across all setups.



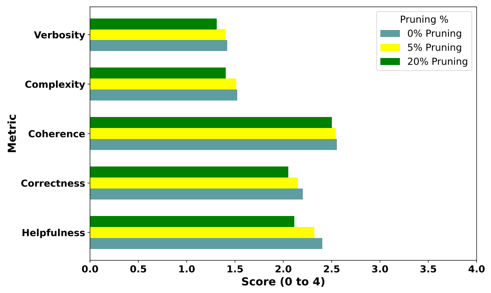
(a) Impact of width pruning on model performance. The figure shows the effect of different pruning percentages on the test dataset, with higher pruning levels reducing scores across all quality metrics, particularly correctness and complexity.



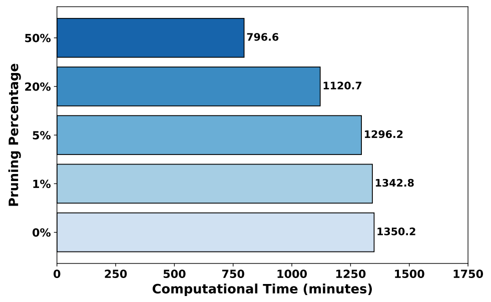
(b) Impact of depth pruning on model performance. The figure illustrates the effect of varying depth pruning percentages, with higher pruning levels leading to a decline in response quality, particularly in coherence and complexity.



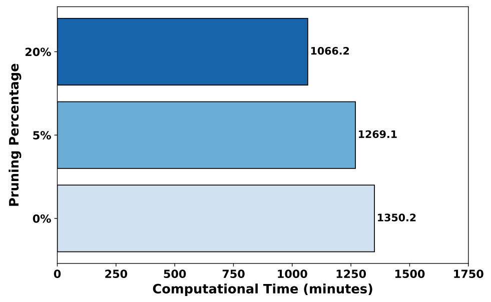
(c) Effect of width pruning on model performance for unseen chemicals. Increasing pruning percentages lead to a decline in response quality, with higher pruning levels significantly affecting correctness and complexity.



(d) Effect of depth pruning on model performance for unseen chemicals. Depth pruning reduces response quality, with higher pruning percentages notably impacting coherence and correctness.

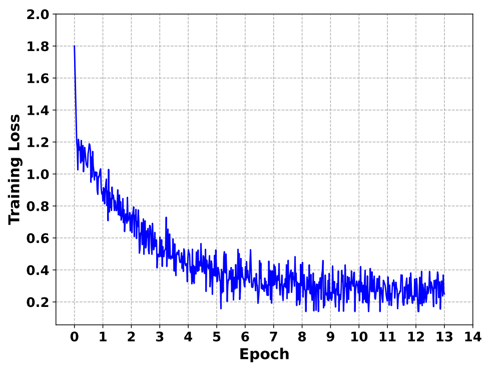


(e) Reduction in computational time with width pruning. Increasing width pruning percentages significantly decreases processing time, with higher pruning levels leading to substantial efficiency gains.

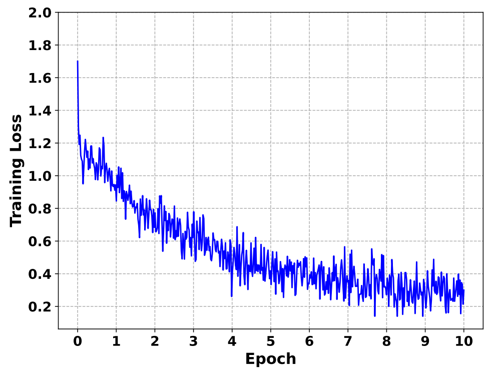


(f) Reduction in computational time with depth pruning. Depth pruning progressively reduces computational cost, with moderate pruning achieving noticeable speed improvements.

Figure 21: Impact of width and depth pruning on model performance and efficiency. The figures illustrate how pruning affects response quality and computational time across both the test dataset and unseen chemical dataset. Higher pruning percentages reduce correctness, coherence, and complexity, with width pruning impacting correctness more and depth pruning affecting coherence. However, pruning also significantly reduces processing time, improving computational efficiency.

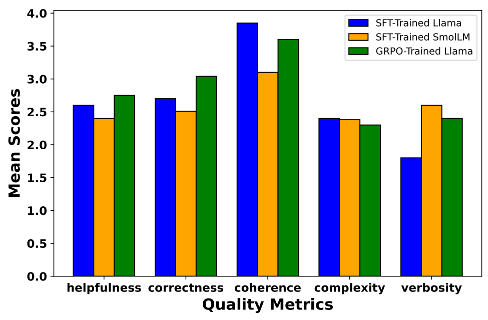


(a) GRPO QA Data Training Loss

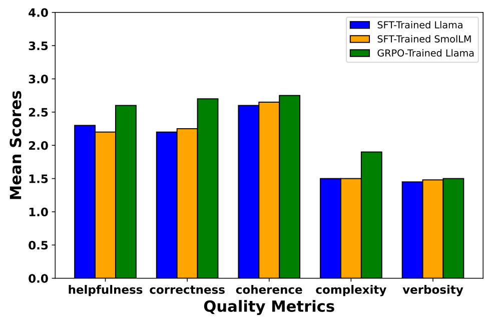


(b) GRPO RAIT Data Training Loss

Figure 22: GRPO Training Loss for 2 datasets



(a) GRPO performance on 1500 QA test dataset



(b) GRPO performance on unseen test dataset

Figure 23: GRPO Trained model performance

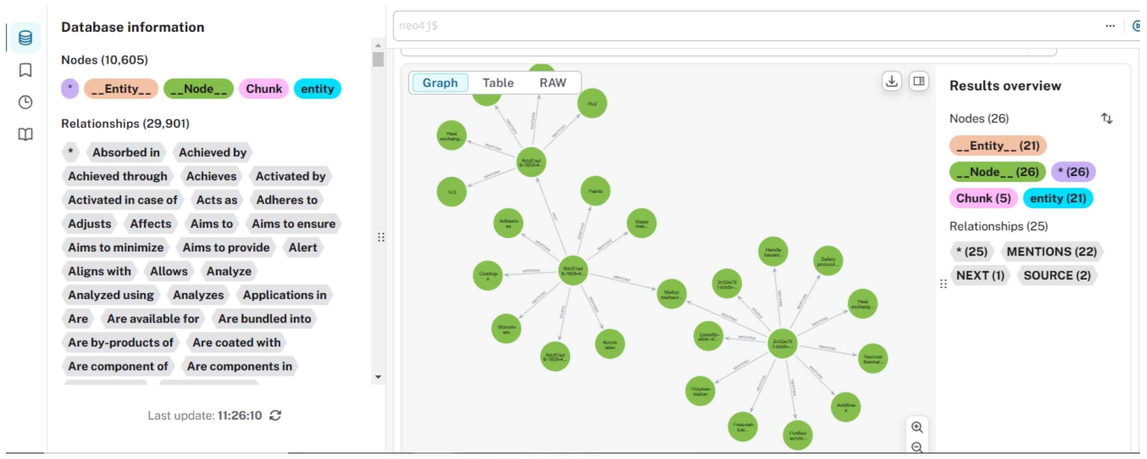


Figure 24: neo4j graph