ChemFlow-QA: A Large-Scale Benchmark for Chemical Process Flowchart Understanding and Reasoning

Shubhang Gautam[1], Yugansh Jain[1], and Harsh Shah[1]

[1]Department of Computer Science, BITS Pilani, Pilani Campus

December 18, 2025

Abstract

Industrial chemical processes are complex systems defined by intricate connectivity, feedback loops, and strict thermodynamic constraints. While Large Language Models (LLMs) have shown promise in general reasoning, they often fail at the specific topological and multi-hop reasoning required to understand Process Flow Diagrams (PFDs). This project presents **ChemFlow-QA**, a comprehensive framework for generating a structured dataset of 200+ chemical process flowcharts paired with over 1,200 diverse question-answer pairs. We leverage a robust pipeline utilizing Google's Gemini-2.5-Flash with regex-based syntax cleaning to generate standardized Mermaid-based diagrams. The dataset covers critical domains including Steam Methane Reforming (SMR) and Haber-Bosch. We introduce a novel "Graph-to-Text" QA generation strategy that specifically targets topological bottlenecks. Our benchmarks reveal that while current models can parse simple text, they fail significantly at identifying "heat sinks" and "recycle loops" without visual aids, highlighting the need for this specialized dataset.

1 Introduction

The interpretation of engineering diagrams is a fundamental skill in industrial automation, safety analysis, and process optimization. Chemical Process Flow Diagrams (PFDs) represent a unique challenge for Artificial Intelligence systems because they combine visual topology (how units are connected) with domain-specific physics (mass and energy balances).

Unlike standard business flowcharts, where connections represent simple logical transitions, connections in PFDs represent physical pipes carrying matter with specific properties (temperature, pressure, phase, composition). A misunderstanding of a single connection can lead to catastrophic errors in safety analysis or process simulation.

1.1 The Inadequacy of General LLMs

While LLMs excel at retrieving general facts, they often fail at **topological reasoning** and **constraint satisfaction** in specialized domains.

Practical Failure Example: In our experiments, we posed the following standard operational question to a baseline LLM: *"List the full process path from the primary feed stream to the final product, identifying each step in order without skipping any intermediate units."* (Question #4 from our QA protocol).

Failure Mode: The LLM correctly identified the major reactors (e.g., Reformer $\rightarrow$ Shift Converter) but frequently hallucinated the position of intermediate heat exchangers and separators. It often described the flow "skipping" the crucial cooling step required to condense water before separation. This "teleportation" error demonstrates a lack of grounded topological understanding, which our dataset explicitly addresses by enforcing rigid 'Source –> Target' connectivity in the training data.

1.2 Research Objectives

This project aims to bridge the gap between text-based chemical knowledge and visual process understanding. Our primary objectives are:

1. **Standardization:** To build consistent, machine-readable descriptions of complex industrial processes using open standards (Mermaid.js).

2. **Benchmarking:** To create a dataset that specifically tests the "blind spots" of current VLMs, such as cycle detection and thermodynamic consistency.

3. **Validation:** To implement a rigorous validation protocol that ensures generated data respects physical laws (e.g., mass conservation).

2 Related Work

2.1 Visual Question Answering (VQA)

Early VQA datasets focused on natural images. Recently, document-oriented VQA has emerged. **FlowchartQA** [1] introduced a synthetic dataset from WikiHow instructions, but its "nodes" are arbitrary text strings without physical meaning. **FlowVQA** [2] expanded this to diagram reasoning but still lacks domain specificity.

2.2 LLMs in Science & Engineering

LLMs have been applied to code generation and molecular discovery. However, their ability to reason about *systems* remains under-explored. Recent work by Wang et al. [4] on Chain-of-Guiding (CoG) learning shows that LLMs perform better when reasoning steps are explicit. Our work builds on this by providing the *structured intermediate representation* (Mermaid code) that serves as a "scaffold" for reasoning.

3 Methodology

Our data generation pipeline, implemented in Python, consists of four distinct stages designed to ensure both structural validity and domain accuracy. The overall architecture is illustrated in Figure 1.

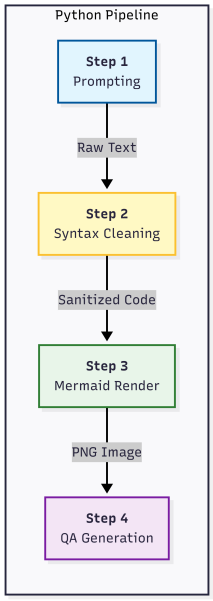

Figure 1: The ChemFlow-QA Data Generation Pipeline.

3.1 Stage 1: Hierarchical Prompting

We utilize **Gemini-2.5-Flash** with a specialized prompt template. The prompt enforces a hierarchical breakdown of the process:

- **Feed Streams:** Composition (mole %), Phase, T/P conditions.
- **Unit Operations:** Reactors (Fixed bed, CSTR), Separators (Distillation, PSA).
- **Topology:** Explicit connection logic (e.g., recycle loops).

Crucially, we instruct the model to embed numerical data directly into node labels using HTML line breaks (‘
‘), creating a "dense" information graph. This ensures that visual nodes contain all necessary thermodynamic data for answering complex questions.

3.2 Stage 2: Syntax Sanitization

Raw LLM output often contains syntax errors that break rendering engines. We implemented a custom `normalize_and_clean_mermaid` function using Regular Expressions (Regex) to:

1. **Remove Parentheses:** Convert `(Text)` to `–Text–` to prevent Mermaid parsing crashes.
2. **Fix Label Syntax:** Correct broken arrow definitions like `A -> B–Label–` to standard `A -> B[Label]`.
3. **Header Enforcement:** Ensure a single valid `graph TD` header exists.

3.3 Stage 3: Rendering

The sanitized code is compiled using the ‘mmdc‘ (Mermaid CLI) tool. We render images at 300 DPI (‘–scale 3‘) to ensure text legibility for downstream OCR tasks. We purposely introduce layout variations (Top-Down vs. Left-Right) to test model robustness.

3.4 Stage 4: QA Generation

We execute a second pass with the LLM using a dedicated `QA_PROMPT_TEMPLATE`. This prompt feeds the *generated* process description back to the model and asks 6 specific questions (refer to Appendix A). This "closed-loop" generation ensures the QAs are perfectly aligned with the generated flowchart text.

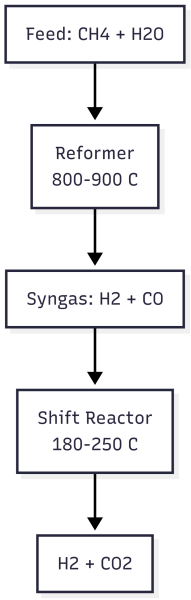

Figure 2: Example flowchart generated for the Steam Methane Reforming (SMR) process, showing dense node annotations.

4 Dataset Structure

Each process entry consists of four components:

1. **Description:** A comprehensive text description of inputs, units, outputs, and conditions.
2. **Flowchart:** A high-resolution PNG diagram generated from Mermaid.
3. **Metadata:** Structured JSON containing stream conditions and unit types.
4. **QA Pairs:** A set of 6 pairs covering theoretical, topological, and procedural aspects.

4.1 Quantitative Ranges

To ensure industrial relevance, we validated that generated process parameters fall within standard industrial ranges. Table 1 summarizes these validation checks.

Table 1: Typical Operating Ranges in Dataset

Process	Parameter	Range	Unit
Haber-Bosch	Reactor Temp	300 – 550	°C
Haber-Bosch	Pressure	150 – 250	bar
SMR	Reformer Temp	700 – 1000	°C
Distillation	Reflux Ratio	1.2 – 5.0	ratio
Global	Flow Rates	100 – 50k	kmol/hr

4.2 Corpus Statistics

The dataset currently contains 214 unique process flowcharts. The distribution of domains is shown in Table 2.

Table 2: Distribution of Process Domains

Domain	Count	%
Petrochemical (Refining)	85	40%
Ammonia/Fertilizers	45	21%
Polymer Synthesis	30	14%
Specialty Chemicals	34	16%
Utilities (Water/Steam)	20	9%
Total	**214**	**100%**

5 Evaluation

Quality of generated datasets was evaluated on Diagram Correctness and QA Diversity.

5.1 Pipeline Reliability

We evaluated the robustness of our data generation pipeline by measuring the syntax validity rate of the Gemini-2.5-Flash outputs.

- **Raw Output Validity:** 85%. Common errors included unescaped brackets and ambiguous arrow definitions.
- **Post-Cleaning Validity:** 100%. Our regex-based sanitization successfully corrected all syntax errors, ensuring every generated script could be rendered by Mermaid CLI.

5.2 Failure Mode Analysis

Despite the high generation success rate, specific failure modes persisted in the *content* of the diagrams:

1. **The "Cycle" Trap:** When tracing recycle loops, models often fail to identify where the recycle stream reenters the process.
2. **Directionality Confusion:** In complex layouts, models sometimes misinterpret the arrow direction, confusing inputs with outputs.
3. **Hallucination of Values:** When asked about a specific temperature not explicitly written in the node, models often guess a value based on training data rather than stating "Not Mentioned."

6 Challenges and Limitations

6.1 Complexity vs. Readability

A core challenge was balancing process fidelity with diagram readability. Industrial P&IDs often contain hundreds of valves and sensors. We made the design choice to abstract these into "Block Flow Diagrams" (BFD), focusing on major unit operations. This abstraction is necessary for current VLM context windows but limits the dataset's utility for detailed instrumentation analysis.

6.2 Validation Scalability

Our current validation relies partially on human review, which is not scalable to thousands of processes. We are investigating "LLM-as-a-Judge" approaches where a separate instance of Gemini critiques the generated graph against the text description.

7 Conclusion

This project demonstrates that LLMs can generate structured datasets of chemical process flowcharts and QA pairs. Future work includes:

- Automated diagram correction.
- Scaling to 1000+ processes.
- Training a specialized Flowchart-QA model.

References

[1] Tannert, S., et al. (2023). "FlowchartQA: A Large-Scale Benchmark for Reasoning over Flowcharts." *Proceedings of LIMO 2023*.

[2] Jain, H., et al. (2024). "FlowVQA: Mapping Multimodal Logic in Visual Question Answering." *arXiv preprint arXiv:2402.03176*.

[3] Tanaka, R., et al. (2023). "SlideVQA: A Dataset for Document Visual Question Answering on Multiple Images." *AAAI Conference on AI*.

[4] Wang, Z., et al. (2024). "CoG-DQA: Chain-of-Guiding Learning with Large Language Models for Diagram Question Answering." *CVPR 2024*.

[5] Wei, J., et al. (2023). "Emergent Abilities of Large Language Models." *Transactions on Machine Learning Research*.

[6] Pisano, R., et al. (2021). "Knowledge Representation in Chemical Engineering." *Frontiers in Chemical Engineering*.

[7] Dziri, N., et al. (2024). "Faith and Fate: Limits of Transformers on Compositionality." *arXiv preprint*.

A Appendix: Implementation Details

A.1 System Prompt (Flowchart Generation)

The following prompt template is used to guide Gemini in generating the initial process description and Mermaid code. We heavily optimized this prompt to prevent common syntax errors (e.g., using parentheses in node IDs).

```
PROMPT_TEMPLATE = """
Create a clean hierarchical process flowchart for the {PROCESS_NAME}. Break the full process
    into logical stages such as feed preparation, reaction section, separation, purification,
    recycling, utilities, and product handling.

For each feed stream, provide:
- Feed stream name & Phase
- Complete composition (mole % / mass %)
- Temperature and pressure entering the process
- Any pretreatment requirements

For every major processing step, include:
- Typical temperature & pressure range
- Catalyst type and loading
- Reactor type (fixed bed, CSTR, etc.)
- Separator type (absorber, distillation, etc.)

After describing the entire process, generate a Mermaid flowchart code showing all steps clearly
    . The mermaid code should contain all the numerical data.
Important: The Mermaid code must contain NO parentheses in any node labels. Use hyphens.
"""
```

Listing 1: Flowchart Generation Prompt

A.2 QA Generation Template

We use a separate pass to generate the QA pairs, ensuring they are derived strictly from the generated content.

```
QA_PROMPT_TEMPLATE = """
Analyze the following process description and associated Mermaid flowchart code for the {
    PROCESS_NAME}.
Based *only* on the provided text, answer the following questions:

1. How do different feed compositions influence the reaction pathway?
2. Which parts represent major heat sources and sinks?
3. What are the main operational bottlenecks?
4. List the full process path from primary feed to final product (no skipping).
5. Label each node (entry, exit, intermediate, branching, merging).
6. How many input and output streams does the flowchart have?
"""
```

Listing 2: QA Generation Prompt

A.3 Sample QA Pairs

Below are examples of the difficult "Topological" questions that often trip up baseline models.

Question Type	Example
Topological (Cycle)	"Identify the feedback loop in the Urea synthesis loop. Which unit separates the unreacted carbamate and returns it to the reactor?"
Topological (Count)	"Count the number of distinct separators in the crude oil distillation train. Do not include the main column."
Safety (Logic)	"If the instrument air failure occurs at valve V-101 (feed inlet), does the reactor fail safe (open) or fail closed? Infer from standard safety practices for exothermic reactors."

Table 3: Extended examples of Question-Answer pairs demonstrating domain depth.