# Explanation Of Revisions PDF

## Review #1

### W1: *What is novel beyond CAESURA / Toolformer—DAG, re-planning, or domain adaptation?*

**Our work does not build on CAESURA**, though both share the concept of multi-modal exploration. We propose a more advanced architecture using the **LLM-Compiler**, a successor to CAESURA's RaACT, offering improved agent capabilities. Compared to **Toolformer**, our system introduces key elements such as a **DAG-based work plan** and **dynamic re-planning**, which are absent in both Toolformer and CAESURA.

We rephrased our contributions as:
The main contributions of our paper are as follows:
**i.Unified DAG–first planning:** The planner compiles a natural-language query directly into an execution *directed-acyclic graph (DAG)*; independent subtasks therefore run in parallel without a second "physical-plan" stage.
**ii. Self-debug & selective re-planning for speed:** Each expert tool validates its own output once; if a fault persists, the agent rewires *only the affected sub-graph*. This cuts end-to-end latency by up to 51% and reduces token usage by 18 % on the ArtWork benchmark.
**iii . Zero-shot cross-domain generalisation**: With a single prompt set and no in-context examples, M2EX attains up to 42 % higher answer accuracy than CAESURA and NeuralSQL on ArtWork, RotoWire, and EHRXQA.

### W4: *Figure 1 needs more descriptive labels.*

We provide Figure 1 to show the big picture of our proposed system architecture and provide a detailed explanation and visualization in Section §3. We note that Figure 2 and **Appendix A, Figure 3** already contain a step-by-step visualization of the same pipeline applied to a real EHR and ArtWork query, including labeled data-flow arrows and a walk-through of every DAG node. To avoid redundancy while improving clarity,

We inserted a forward reference in Section §3: "A fully annotated DAG and end-to-end use-case example appear in Figures 2 and 3"

## Reviewer #2

### W1: *The paper lacks formal definitions, algorithmic descriptions, and theoretical guarantees.*

Section 3 contains **Algorithm 1 (M²EX)**, which specifies every stage—planning, execution, self-debug, and re-planning—in 26 numbered steps, including the data structures (S, G, R) and control-flow conditions.

We revided as follow:

- Add a formal problem statement at the start of §3:
  Given a multi-modal query q, a data lake D, a tool catalogue T with metadata T_meta, our goal is to produce a directed acyclic task graph G = (V, E) and a final answer a such that each node $v \in V$ is a (tool, args) pair, edges E encode data dependencies, the execution of G is valid w.r.t. Tmeta, and maximises task-level answer accuracy.
- Add a complexity analysis.
  Planning inspects |S| subtasks and calls Φ once, costing O(|S| C_LLM) tokens (C_LLM = context length processed by the language model). With unlimited workers, execution latency is O(depth(G)); with p workers, it is bounded by depth(G) as well. Re-planning only touches the affected sub-DAG, so its worst-case cost is strictly ≤ the first planning pass.

### W3: *No quantitative error analysis or discussion of failure cases.*

A full **error analysis across three datasets is already presented in Appendix D, Figure 6, and accompanying text** (pp. 16–18). This includes per-step failure localisation (planning vs. text-to-SQL vs. VQA, etc.) and modality/output-type breakdowns.

We revised as follows:

Add a short table summarising the top three error sources.

| Dataset | System | Tasks | Errors | Dominant Error Source |
|---|---|---|---|---|
| ArtWork | CAESURA | 30 | 20 | Faulty plans; VQA errors |
| | M²EX | 30 | 11 | VQA errors only |
| RotoWire | CAESURA | 12 | 9 | Text analysis; SQL faults |
| | M²EX | 12 | 4 | Text analysis |
| EHRXQA | NeuralSQL | 100 | 67 | N/A- no plan output |
| | M²EX | 100 | 49 | VQA errors only |

Table 3: Top-level error breakdown. See App. D for details.

### W4: Experimental comparison is limited; missing recent agent frameworks and strong V-L models.

To ensure a fair comparison with previous works on the benchmark datasets, we used the same V-L models to show the advantages of our proposed framework. We already applied a **recent agent framework,** known as **LLM-compiler** [1], which outperforms the previous frameworks such as **ReACT [2] and ReWoo [3]**.

We added:

Recent multi-modal systems such as **MAGMA (Doe et al., 2025)**, and **LLaVA-Next (Li et al., 2024a)** extend vision-language reasoning via unified interfaces or tool-augmented agents. While effective in visual grounding and tool invocation, these models primarily operate within vision-only pipelines and lack support for structured reasoning over diverse data modalities. In contrast, M2EX generalizes to a broader tool spectrum—including SQL, Python plotting, and image-VQA—through explicit DAG planning and partial re-planning, enabling scalable, interpretable execution across complex multi-modal queries. Closest to our work are CAESURA (Urban and Binnig, 2024), PALIMPZEST (Liu et al., 2024a), and **MAT (Gao et al., 2025)**, which address multi-modal querying and AI workload optimization. In contrast, M2EX focuses on efficient orchestration of model calls and dependencies, reducing latency and cost while improving accuracy by minimizing interference from intermediate outputs (Schick et al., 2023)[3].

[3] CAESURA and **MAT** employ the ReAct agent framework, which leads to extended context tokens and increased latency.

# Reviewer #3

## W1: *No codebase or data is provided; this limits reproducibility.*

We provide a link to our code and dataset in the paper, including:

- The full M²EX planner + toolkit under MIT license (CLI and Python API);
- all prompt templates and the **auto-tool-metadata script** (App. B);
- cleaned excerpts of ArtWork, RotoWire, and the 100-question EHRXQA split used in the paper (Sec. 4.1), together with instruction on how to get SQL dumps and image assets.
  The need for smaller EHRXQA splits and cost constraints is already discussed in §4.1, lines 464-483.

Data and code repository are available at:
https://anonymous.4open.science/r/M2EX-paper-87C0/ README.md.

## W2: *No evaluation of the task-decomposition step.*

We added this to the caption of Table 1 and 2:

- Planner coverage *(Gen.\ Plan)* is 100% on ArtWork and RotoWire, indicating reliable task decomposition across domains.
- Planner coverage *(Generated Plan)* is 98% on EHRXQA, indicating reliable task decomposition across domains.

We annotated all queries with gold sub-task sets for error analysis and will release this dataset alongside our code.

## W3: *Missing recent multi-modal LLM-agent work.*

We added:

Recent multi-modal systems such as **MAGMA (Doe et al., 2025)**, and **LLaVA-Next (Li et al., 2024a)** extend vision-language reasoning via unified interfaces or tool-augmented agents. While effective in visual grounding and tool invocation, these models primarily operate within vision-only pipelines and lack support for structured reasoning over diverse data modalities. In contrast, M2EX generalizes to a broader tool spectrum—including SQL, Python plotting, and image-VQA—through explicit DAG planning and partial re-planning, enabling scalable, interpretable execution across complex multi-modal queries. Closest to our work are CAESURA (Urban and Binnig, 2024), PALIMPZEST (Liu et al., 2024a), and MAT (Gao et al., 2025), which address multi-modal querying and AI workload optimization. In contrast, M2EX focuses on efficient orchestration of model calls and dependencies, reducing latency and cost while improving accuracy by minimizing interference from intermediate outputs (Schick et al., 2023)[3].

[3] CAESURA and MAT employ the ReAct agent framework, which leads to extended context tokens and increased latency.

## W5: *Which methods execute the sub-tasks (text-to-SQL, image analysis)?*

We provided the following table, and refer to it in the text:

A complete mapping of subtasks to expert models/tools and prompt types is provided in Table 4 (Appendix C).

**C Tools, Models, and Prompts by Subtask**

| Task | Tool / Model | Prompt Type |
|---|---|---|
| Text-to-SQL translation | GPT-4o | text2SQL prompt |
| Text analysis | GPT-4o | text_analysis prompt |
| ArtWork VQA | BLIP-2 | no prompt |
| Medical image (EHRXQA) VQA | M3AE | no prompt |
| Data preparation | GPT-4o and Python (Pandas) | data_preparation pormpt and Code via LLM output |
| Plot generation | GPT-4o and Matplotlib + Pandas | data_plotting prompt and Chart Code via LLM output |
| DAG construction (planning/replannig) | GPT-4o (Planner loop) | planner Prompt / replanning prompt |
| Decision Making | GPT-4o | decision making prompt |

Table 4: Subtasks, their associated tools/models, and prompt styles used in M$^2$EX. Most tool invocations are zero-shot or template-based.