TOWARD DOMAIN-SPECIFIC INNOVATION DISCOVERY IN COMPUTER SCIENCE RESEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in Large Language Models (LLMs) have substantially improved scientific hypothesis formulation and research idea generation. However, existing approaches often produce superficial recombinations of prior solutions, while external literature resources rarely provide the fine-grained, structured evidence required to develop technically coherent ideas for complex and interdisciplinary research problems. To address these limitations, we present *CSMuse*, an evidence-grounded idea-generation system that leverages cross-domain knowledge. *CSMuse* incorporates three key designs: (1) domain-specific scientific knowledge construction, (2) evidence-grounded idea generation and validation, and (3) cross-domain multi-agent ideation and refinement. Evaluations across multiple LLM backbones and research domains demonstrate that *CSMuse* consistently generates high-quality, novel ideas and outperforms state-of-the-art automated scientific discovery systems.

1 INTRODUCTION

As research problems in computer science become increasingly complex and interdisciplinary, researchers need tools that can help them explore vast design spaces without compromising technical validity. Meanwhile, the rapid development of LLMs has enabled notable progress in scientific hypothesis and idea generation. For example, AIScientist Lu et al. (2024) uses LLMs with prompt templates to generate research directions. AUTODISCOVERY Agarwal et al. (2025) elicits prior and posterior beliefs about hypotheses via sampling, and uses surprisal as a reward within an MCTS procedure to balance exploration and exploitation in search of surprising discoveries. Other studies incorporate external knowledge to support idea generation. For instance, SCIMON Wang et al. (2024b) trains a language model Raffel et al. (2020) on an open-source paper dataset Lo et al. (2020) to generate ideas and iteratively refine their novelty. ResearchAgent Baek et al. (2025) leverages citation-graph-based retrieval to provide LLMs with relevant and potentially cross-domain scientific knowledge to generate and iteratively refine research ideas. VIRSCI Su et al. (2025) models multiple agents as collaborating researchers, each supported by an additional corpus of paper abstracts. Despite this progress, existing approaches remain insufficient for complex, interdisciplinary computer science research for three main reasons.

LLM-generated ideas lack explicit evidential grounding. Standalone LLM-based idea generation mainly draws from large-scale knowledge stored in model parameters, making it difficult to trace the evidence behind a generated idea or verify its technical validity. Consequently, the generated outputs are often recombinations of memorized patterns. This limitation is particularly problematic in areas such as computer networks and systems, where viable ideas require precise architectural reasoning and technically consistent designs.

Open-source literature resources lack structured knowledge relationships. Existing open-source resources provide broad bibliographic corpora Lo et al. (2020); Morris et al. (2023), full-text papers Association for Computing Machinery (2026); IEEE (2026); USENIX Association (2026); Cornell University (2026), or curated metadata through platforms such as Semantic Scholar Allen Institute for AI (2026) and OpenAlex Priem et al. (2022). However, they generally do not explicitly capture the fine-grained relationships among research domains, problems, motivations, and design mechanisms. This limits their ability to provide the structured evidence needed for systematic idea generation.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Cross-domain knowledge deserves more than combining ideas. Existing approachesBaek et al. (2025); Su et al. (2025) use multiple agents or combine ideas from different sources to increase diversity. However, simply aggregating independently generated ideas does not ensure meaningful cross-domain transfer and may instead produce incoherent or technically incompatible combinations. This reduces the technical coherence and practical feasibility of the resulting ideas.

To address these limitations, we present *CSMuse*, an idea-generation system designed for complex and interdisciplinary computer science research. We construct a scientific discovery dataset from premier conferences across multiple computer science domains. Moreover, we use knowledge-graph-based evidence to ground and verify generated ideas. Finally, we employ an LLM-driven, cross-domain multi-agent framework for idea generation and refinement. The three contributions are detailed as follows.

- **Domain-specific scientific knowledge construction.** We collect papers from premier conferences across multiple computer science domains. We then use LLMs to extract each paper's core research content, producing a consistently formatted dataset that preserves detailed information about research problems and system designs.
- **Evidence-grounded idea generation and validation.** We organize the dataset into domain-specific knowledge graphs and retrieve relevant knowledge as explicit evidence for each generated idea. We further assess whether the final idea is adequately supported by the source literature by measuring its textual similarity to the retrieved evidence.
- **Cross-domain multi-agent ideation and refinement.** We mimic the collaborative scientific reasoning process through a multi-agent workflow. Specifically, a generation agent first proposes a seed idea, and a critic agent then identifies its limitations and suggests directions for revision. Guided by this critique, domain-specific agents retrieve relevant evidence from their respective knowledge graphs and independently refine the idea from different disciplinary perspectives. Finally, an integration agent synthesizes these revisions into a coherent final idea.

Evaluation Results. We implement a prototype of *CSMuse* and evaluate the quality of its generated ideas across multiple LLM backbones and research domains. The results demonstrate that *CSMuse* consistently outperforms two state-of-the-art automated scientific discovery systems, AI Scientist Lu et al. (2024) and VIRSCI Su et al. (2025). Our ablation studies further validate the effectiveness of its evidence-grounding and cross-domain refinement components in complex and interdisciplinary computer science research.

2 RELATED WORK

2.1 LITERATURE-BASED DISCOVERY

Literature-Based Discovery (LBD) aims to automatically uncover new knowledge from existing scientific literature Swanson (1986). Its classical formulation originates from Swanson's ABC co-occurrence model Swanson & Smalheiser (1997), which infers implicit relationships from explicit textual evidence (*e.g.*, "A implies B" and "B implies C"). More recent approaches use link-prediction models Ghafarollahi & Buehler (2025); Wang et al. (2019); Sybrandt et al. (2020) to formulate scientific hypotheses as previously unobserved relationships between concepts.

2.2 LLM-BASED SCIENTIFIC DISCOVERY

Recent studies have leveraged LLMs to generate scientific hypotheses and research ideas Wang et al. (2024b); Lu et al. (2024); Agarwal et al. (2025). These approaches use the broad knowledge and reasoning capabilities of LLMs to explore large hypothesis spaces, while some additionally incorporate literature corpora or search procedures to improve novelty and relevance. However, their generated ideas are often grounded only implicitly in model parameters or coarse-grained literature representations, providing limited traceable evidence for technically complex computer science research.

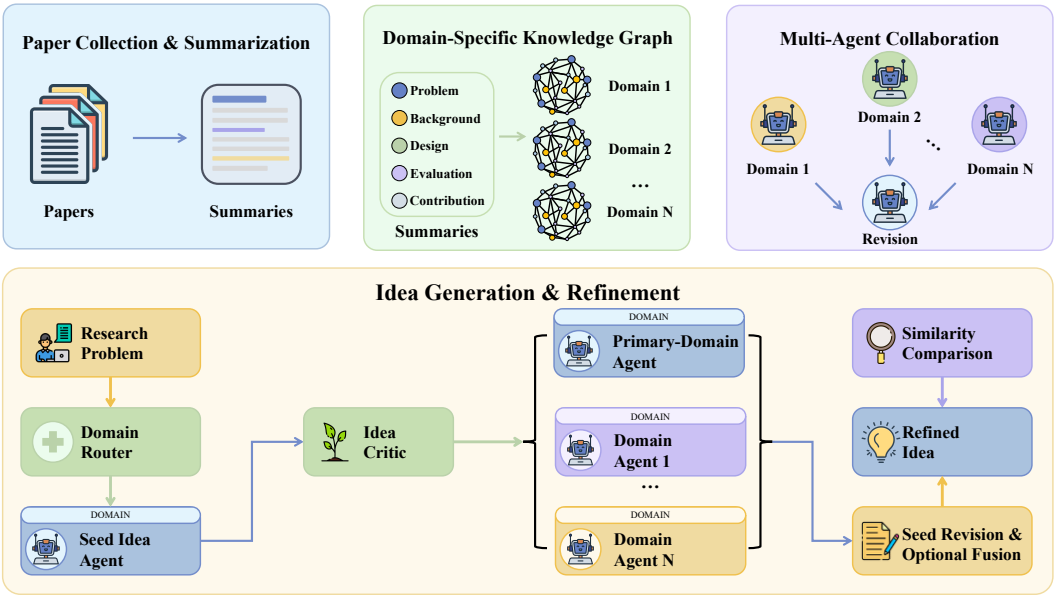


Figure 1: Overview of *CSMuse*. It summarizes papers from premier conferences into a structured dataset, constructs domain-specific knowledge graphs, retrieves evidence relevant to a user-specified research problem, and generates and refines ideas through cross-domain multi-agent collaboration.

2.3 Multi-Agent Scientific Discovery

Other systems introduce agentic or multi-agent workflows for scientific idea generation Baek et al. (2025); Su et al. (2025). By assigning different roles to multiple agents, these systems support iterative discussion, critique, and refinement. Nevertheless, existing approaches generally do not organize scientific knowledge by domain or explicitly reason about how mechanisms should be adapted across domains.

3 Overview

This section provides an overview of *CSMuse*. As shown in Figure 1, the three components in the upper part correspond to our key designs: paper collection and summarization for the first contribution, domain-specific knowledge graph for the second contribution, and multi-agent collaboration for the third contribution. The lower part illustrates the end-to-end workflow for idea generation and refinement in *CSMuse*.

Paper Collection & Summarization. We collect papers published from 2021 to 2025 at premier conferences across multiple computer science domains, following the ACM Computing Classification System ACM (2026) to organize these domains. We then use LLMs to summarize and normalize each paper into five structured fields: $Problem$, $Background$, $Design$, $Evaluation$, and $Conclusion$. This process provides detailed yet concise, semantically consistent, and domain-specific knowledge for idea generation. Compared with raw PDFs, which are unstructured and semantically heterogeneous, or abstracts, which often omit implementation details, this structured representation better preserves mechanism-level design information.

Domain-Specific Knowledge Graph. We further condense the structured paper summaries and use them to construct a separate knowledge graph for each domain. These domain-specific knowledge graphs enable agents to retrieve relevant, traceable evidence when generating and refining research ideas.

Domain-Aware Multi-Agent Collaboration. We instantiate a specialized agent for each domain. Starting from a seed idea generated by the primary-domain agent, these agents retrieve supporting knowledge from their respective domain-specific graphs and independently revise the idea from

different disciplinary perspectives. An integration agent then synthesizes the resulting revisions into a coherent, evidence-grounded idea.

Idea Generation & Refinement. The workflow begins with a user-specified research problem. A domain router analyzes the problem keywords, selects the most relevant primary domain, and invokes the corresponding domain agent to generate a seed idea. A critic agent then evaluates the seed idea, identifies its key defects, and specifies the evidence needed to address them. Guided by this critique, domain-specific agents retrieve relevant evidence from their respective knowledge graphs and propose mechanism-level revisions. An integration agent selectively combines compatible revisions into a coherent idea. Finally, *CSMuse* compares the revised idea with the paper-summary dataset and modifies non-essential mechanisms that substantially overlap with prior work, producing the final refined idea.

4 Summarization & Knowledge Graph Construction

This section describes the construction of the structured paper-summary dataset (§ 4.1) and the corresponding domain-specific knowledge graphs (§ 4.2).

4.1 Paper Summarization

To build a high-quality dataset for idea generation, we collect papers published between 2021 and 2025 at premier conferences spanning research domains in the ACM Computing Classification System (CCS). For each domain, we select two or three representative conferences. The summarization pipeline consists of two stages.

Semantic Cleaning. At the document level, we first use MinerU Wang et al. (2024a) to convert each PDF into Markdown, enabling LLMs to process the complete paper content in a more structured and accessible format. At the domain level, we use LLMs to identify the research area of each paper. Because similar areas may be described using different terminology or levels of granularity, unconstrained labels can be inconsistent across papers. We therefore restrict the generated labels to third-level categories in the ACM CCS taxonomy, ensuring consistent terminology and granularity throughout the dataset.

Paper Summarization. We then use LLMs to summarize each paper into five structured components: *Problem* (paper identifier and research problem), *Background* (domain, subdomain, motivation, assumptions, and related methods), *Design* (key idea, mechanism, and system architecture), *Evaluation* (metrics, testbed, baselines, and results), and *Conclusion* (limitations, failure modes, deployment constraints, and novelty clues). We normalize the resulting summaries into JSON format. The detailed summarization prompt is provided in Appendix A.1.

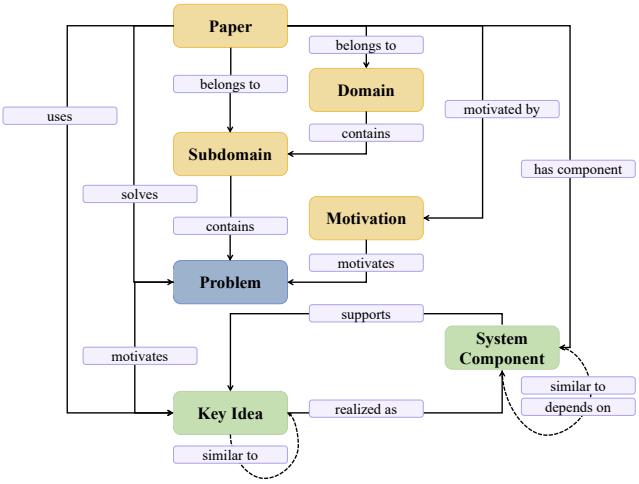


Figure 2: Schema of a domain-specific knowledge graph. Solid edges encode structural and causal relationships: a *Paper* belongs to a *Domain* and *Subdomain*, solves a *Problem*, is motivated by a *Motivation*, uses a *Key Idea*, and has a *System Component*; a *Domain* contains *Subdomains*; a *Subdomain* contains *Problems*; a *Motivation* motivates a *Problem*; a *Problem* motivates a *Key Idea*; and a *Key Idea* is realized as a *System Component*, which supports the idea and may depend on other components. Dashed edges represent semantic similarity between *Key Ideas* and between *System Components*.

4.2 Domain-Specific Knowledge Graph

To explicitly capture logical connections within and across papers, we construct a separate paper-level knowledge graph for each research domain using the structured paper-summary dataset. Each graph contains seven entity types: $Paper$, $Problem$, $Domain$, $Subdomain$, $Motivation$, $Key\ Idea$, and $System\ Component$. For each summary, the $Paper$ entity corresponds to the paper identifier, while the $Problem$ entity is extracted from the $Problem$ field. The $Domain$, $Subdomain$, and $Motivation$ entities are extracted from $Background$, whereas $Key\ Idea$ and $System\ Component$ are extracted from $Design$.

We define fifteen types of directed relationships among these entities, as illustrated in Figure 2, and use GraphRAG Edge et al. (2024) to construct the domain-specific graphs. These graphs make the relationships among research domains, problems, motivations, key ideas, system components, and papers explicit, thereby supporting structured and traceable evidence retrieval during idea generation.

5 Domain-specific Multi-Agent Idea Discovery

This section presents the domain-specific multi-agent workflow used by ***CSMuse*** for research idea discovery.

5.1 Seed Idea Generation

During seed idea generation, ***CSMuse*** takes the research context $\mathcal{B} = \mathcal{P} \oplus \mathcal{D}$ as input, consisting of (1) a research problem $\mathcal{P}$ and its description and (2) a research domain $\mathcal{D}$. A domain router extracts keywords from the research problem and maps it to the most relevant ACM CCS ACM (2026) domain based on their occurrence frequencies. ***CSMuse*** then incorporates the resulting research context into the seed-idea-generation prompt provided in Appendix A.3. The agent associated with the selected domain uses this prompt to generate a seed idea $\mathcal{I}_s$.

5.2 Multi-Agent Collaboration

The multi-agent collaboration stage consists of two components: idea critique and domain-agent collaboration. The corresponding prompt is provided in Appendix A.4.

Idea Critic. After the seed idea is generated, the critic agent takes the research-idea context $\mathcal{I} = \mathcal{P} \oplus \mathcal{I}_s$ as input, consisting of (1) the research problem $\mathcal{P}$ and its description and (2) the seed idea $\mathcal{I}_s$. It evaluates the seed idea along five dimensions: problem fit, technical correctness, validity of assumptions, feasibility, and adequacy of the proposed evaluation. The critic then prioritizes the identified defects, specifies the evidence required to address each defect, and produces critique guidance $\mathcal{C}$.

Domain Agents Collaboration. We construct the retrieval context $\mathcal{K} = \mathcal{I} \oplus \mathcal{C}$ by combining the research-idea context $\mathcal{I}$ with the critique guidance $\mathcal{C}$. For each domain d, the corresponding knowledge graph uses $\mathcal{K}$ to retrieve relevant evidence $\mathcal{E}_d$. The domain agent then takes $\mathcal{K} \oplus \mathcal{E}_d$ as input and produces a domain-specific supplement $\mathcal{SUP}_d$. Each supplement provides a concise mechanism-level revision that addresses the identified defects using evidence retrieved from the corresponding domain.

5.3 Idea Refinement

The idea-refinement stage consists of two components: seed revision with optional fusion and similarity-based novelty refinement. The corresponding prompt is provided in Appendix A.5.

Seed Revision & Optional Fusion. The fusion agent takes the research-idea context $\mathcal{I}$, the critique guidance $\mathcal{C}$, and the domain-specific supplements $\{\mathcal{SUP}_d\}$ as input. It evaluates each supplement for relevance, evidential support, and compatibility with the seed idea, selectively incorporating only those that address the identified defects without introducing technical inconsistencies. The agent then synthesizes the selected supplements into a revised idea $\mathcal{I}_r$.

Similarity-Based Novelty Refinement. We next compare the revised idea $\mathcal{I}_r$ with the summarized papers in our dataset using SPECTER Cohan et al. (2020). For each paper, we first compute the cosine similarity between its $Problem$ field and the target research problem $\mathcal{P}$. If this similarity exceeds $t_{\text{problem}} = 0.5$, we further compare the paper's $Mechanism$ field with the mechanism proposed in $\mathcal{I}_r$. When the mechanism similarity exceeds $t_{\text{mechanism}} = 0.8$, we record the overlapping content as historical evidence $\mathcal{E}_h$. A novelty agent then takes $\mathcal{P}$, $\mathcal{I}_r$, and $\mathcal{E}_h$ as input, identifies non-essential mechanisms that substantially overlap with prior work, and revises only those components. This process produces the final refined idea $\mathcal{I}_f$ while preserving the technically necessary parts of the design.

6 Evaluation

6.1 Evaluation Settings

Dataset. We evaluate *CSMuse* and the baseline methods using openly available papers from the ACM Digital Library Association for Computing Machinery (2026), USENIX proceedings USENIX Association (2026), and OpenReview OpenReview (2026). Following the ACM CCS ACM (2026) taxonomy, we cover three research domains: Computing Methodologies (CM), Networks (Net), and Computer Systems Organization (CSO). For each domain, we select two or three representative conferences and collect papers published between 2021 and 2026, as summarized in Appendix B.1. The resulting dataset contains 968 papers in Computing Methodologies, 879 papers in Networks, and 583 papers in Computer Systems Organization. We partition the dataset chronologically into two subsets:

- Historical papers ($Paper_{\text{past}}$). Papers published between 2021 and 2025 are converted into structured summaries, used to construct domain-specific knowledge graphs, and treated as the historical corpus for measuring the similarity of generated ideas to prior work.
- Contemporary papers ($Paper_{\text{con}}$). Papers published in 2026 are converted into structured summaries and used both to derive 150 evaluation research problems and to measure the similarity between generated ideas and contemporary work.

The paper summaries, knowledge graphs, and evaluation research problems are generated using *DeepSeek-V4-Pro*. The corresponding prompts are provided in Appendix B.1.

Evaluation Metrics. We evaluate generated ideas in terms of both novelty and overall quality. Following the evaluation methodology of VirSci Su et al. (2025), we divide the corpus into historical papers published from 2021 to 2025 and contemporary papers published in 2026. Based on this temporal split, we define two embedding-based metrics. (1) *Historical Dissimilarity (HD)* is defined as the average cosine similarity between the generated-idea embedding and the embeddings of its three most similar paper summaries in $Paper_{\text{past}}$. A lower HD therefore indicates less overlap with prior work and greater potential novelty. (2) *Contemporary Dissimilarity (CD)* is the average cosine similarity between the generated-idea embedding and the embeddings of its five nearest paper summaries in $Paper_{\text{con}}$. A higher CD indicates stronger similarity to independently published contemporary work, suggesting that the generated idea captures plausible and timely research directions.

Embedding-based metrics alone may underestimate novelty because papers within the same domain often share terminology, even when their underlying mechanisms differ substantially. We therefore additionally employ an *LLM-as-a-Judge (LJ)* metric to assess each idea along six dimensions: problem fit, mechanism specificity, technical soundness, evaluation quality, novelty, and feasibility. Each dimension is scored on a five-point scale.

Finally, we combine the similarity-based metrics and the LLM judgment into an *Overall Idea Score (OIS)*:

$$\text{OIS} = \frac{\text{LJ}}{5} \times \frac{\text{CD}}{\text{HD}}.$$

Models. We evaluate *CSMuse* using models from two representative LLM providers: *GPT-5.5* from OpenAI (knowledge cutoff: December 2025) OpenAI (2026) and *DeepSeek-V4-Pro* from DeepSeek

(a) Ideas generated by *DeepSeek-V4-Pro*.

	LJ/5			OIS		
	CM	Net	CSO	CM	Net	CSO
AI Scientist	0.7571	0.7707	0.7716	0.74591	0.7599	0.7672
VIRSCI	0.7664	0.7613	0.7827	0.7562	0.7515	0.7763
CSMuse	0.8727	0.8700	0.8433	0.8611	0.8644	0.8386

(b) Ideas generated by *GPT-5.5*.

	LJ/5			OIS		
	CM	Net	CSO	CM	Net	CSO
AI Scientist	0.6527	0.6499	0.6347	0.6416	0.6395	0.6271
VIRSCI	0.6102	0.6371	0.5996	0.6009	0.6252	0.5939
CSMuse	0.8020	0.8260	0.8197	0.7907	0.8179	0.8172

Table 1: Idea-quality comparison of *CSMuse* and baseline methods across research domains and LLM backbones.

	LJ/5			OIS		
	CM	Net	CSO	CM	Net	CSO
Vanilla LLM	0.5833	0.5887	0.5647	0.5741	0.5824	0.5624
LLM with Retrieval	0.8080	0.8140	0.7847	0.7987	0.8061	0.7808
LLM with Refinement	0.8507	0.8340	0.8200	0.8431	0.8267	0.8167
LLM with Retrieval and Refinement	0.8500	0.8447	0.8173	0.8402	0.8392	0.8126
CSMuse	0.8727	0.8700	0.8433	0.8611	0.8644	0.8386

Table 2: Impact of *CSMuse* components on idea quality across research domains and LLM backbones.

(knowledge cutoff: April 2026) DeepSeek (2026). This selection allows us to examine whether *CSMuse* performs consistently across different model families.

Baselines. We compare *CSMuse* with two SOTA systems for automated scientific discovery: the single-agent AI Scientist Lu et al. (2024) and the multi-agent VIRSCI Su et al. (2025). Because these systems employ different generation pipelines, we align their experimental settings as closely as possible and evaluate all generated ideas using the same embedding-based metrics and LLM-as-a-Judge protocol.

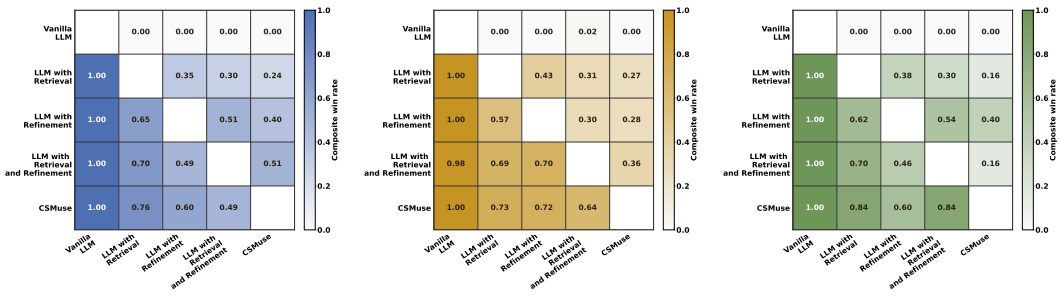
6.2 COMPARISONS WITH BASELINES

In this experiment, we evaluate the quality of ideas generated by *CSMuse* across different research domains and LLM backbones. As shown in Table 1, *CSMuse* consistently outperforms the baselines under both the Overall Idea Score (OIS) and LLM-as-a-Judge (LJ) metrics. Detailed LJ results are provided in Appendix B.2. These results demonstrate that evidence grounding and cross-domain collaboration improve the novelty and overall quality of generated ideas. The improvements are consistent across both *GPT-5.5* and *DeepSeek-V4-Pro*, indicating that the effectiveness of *CSMuse* generalizes across different underlying models.

6.3 ABLATION STUDY

To isolate the contribution of each component in *CSMuse*, we compare the complete system against four ablated variants. (1) *Vanilla LLM* uses a single agent and a standard prompt without retrieved evidence or iterative refinement. (2) *LLM with Retrieval* uses a single agent augmented with evidence retrieved from the primary domain. (3) *LLM with Refinement* employs multi-agent refinement without knowledge-graph evidence and restricts to the primary domain. (4) *LLM with Retrieval and Refinement* combines retrieval and refinement but restricts both to the primary domain, thereby removing cross-domain knowledge transfer. We use *DeepSeek-V4-Pro* to generate ideas for problem sets from the three research domains. The resulting LLM-as-a-Judge (LJ) and Overall Idea Score (OIS) values are reported in Table 2. Detailed LJ results are provided in Appendix B.2.

Overall, *CSMuse* consistently generates higher-quality ideas than the ablated variants across the evaluated research domains. Adding retrieved evidence and iterative refinement progressively improves both LJ and OIS, while the complete system achieves the strongest performance. These results demonstrate the complementary benefits of evidence grounding and cross-domain collaboration.



(a) Win Rate in Computing Methodologies

(b) Win Rate in Networks

(c) Win Rate in Computing Systems Organization

Figure 3: Pairwise win rates of the ablated variants across research domains, excluding ties. Retrieval and Refinement slightly outperforms *CSMuse* in Computing Methodologies (51% vs. 49%). Refinement alone also outperforms Retrieval and Refinement in Computing Methodologies (51% vs. 49%) and Computer Systems Organization (54% vs. 46%).

We further conduct pairwise comparisons in which an LLM judge selects the stronger idea produced for each problem, as shown in Figure 3. For cross-domain knowledge transfer, we compare *CSMuse* with Retrieval and Refinement, which excludes cross-domain knowledge; the latter slightly outperforms *CSMuse* in Computing Methodologies (51% vs. 49%). For evidence grounding, we compare Retrieval and Refinement with Refinement alone; Refinement alone performs better in Computing Methodologies (51% vs. 49%) and Computer Systems Organization (54% vs. 46%). These mixed results motivate a more fine-grained analysis of both components. Specifically, we evaluate evidence grounding using the similarity-based CD/HD component, which captures how retrieved knowledge is reflected in the generated ideas, and evaluate cross-domain knowledge transfer using the LLM-as-a-Judge score, which measures improvements in overall technical quality. The following sections present these component-specific analyses.

6.4 Evidence-Grounding Study

To quantify the contribution of evidence grounding, we compare *CSMuse* with the *LLM with Refinement* variant on each evaluation problem. This comparison isolates the effect of retrieved knowledge while retaining the same refinement process. For each idea pair $\langle I_i, I_j \rangle$, where I_i is generated by *CSMuse* and I_j is generated by *LLM with Refinement*, we compute the differences in Overall Idea Score and similarity ratio as

$$\Delta_{\text{OIS}} = S_{\text{OIS}}(I_i) - S_{\text{OIS}}(I_j), \qquad \Delta_{CD/HD} = S_{CD/HD}(I_i) - S_{CD/HD}(I_j).$$

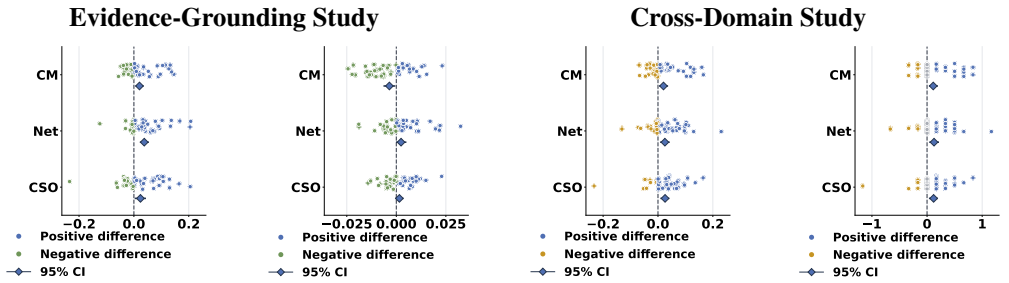
The results are summarized in Figure 4. Points to the right of zero indicate that *CSMuse* performs better on the corresponding research problem, whereas points to the left favor the ablated variant.

In terms of Overall Idea Score (Figure 4a), *CSMuse* achieves an average win rate of 64.14% and an average improvement of $+0.0275$. Across individual domains, the win rates and average improvements are 60.42% and $+0.0204$ for Computing Methodologies, 72.00% and $+0.0377$ for Networks, and 59.57% and $+0.0239$ for Computer Systems Organization, respectively.

We further isolate the similarity-based component, CD/HD, to examine whether the retrieved evidence is reflected in the generated ideas (Figure 4b). Overall, *CSMuse* achieves a win rate of 51.03% and an average improvement of $+0.0002$. For Computing Methodologies, Networks, and Computer Systems Organization, the corresponding win rates are 41.67%, 60.00%, and 51.06%, with average differences of -0.0035, $+0.0024$, and $+0.0015$, respectively. These results suggest that evidence grounding is particularly beneficial for the Networks and Computer Systems Organization domains, where idea generation often requires complex architectural and system-level reasoning.

6.5 Cross-domain Study

To quantify the contribution of cross-domain knowledge transfer, we compare *CSMuse* with the *LLM with Retrieval and Refinement* variant on each evaluation problem. This variant retains evidence retrieval and iterative refinement but restricts both to the primary domain, allowing us to isolate



(a) Overall Idea Score **(b)** Similarity Ratio **(c)** Overall Idea Score **(d)** LLM-as-a-Judge Score

Figure 4: Results of the evidence-grounding and cross-domain studies. Panels (a) and (b) compare *CSMuse* with LLM with Refinement, whereas panels (c) and (d) compare *CSMuse* with LLM with Retrieval and Refinement. Each panel reports the mean score difference across paired ideas and its 95% confidence interval.

the benefits of incorporating knowledge from other domains. For each idea pair $\langle I_i, I_k \rangle$, where I_i is generated by *CSMuse* and I_k is generated by the primary-domain-only variant, we compute the differences in Overall Idea Score and LLM-as-a-Judge score as

$$\Delta_{\mathrm{OIS}} = S_{\mathrm{OIS}}(I_i) - S_{\mathrm{OIS}}(I_k), \qquad \Delta_{\mathrm{LJ}} = S_{\mathrm{LJ}}(I_i) - S_{\mathrm{LJ}}(I_k).$$

The results are summarized in Figure 4. Points to the right of zero indicate that *CSMuse* performs better on the corresponding research problem, whereas points to the left favor the primary-domain-only variant.

In terms of Overall Idea Score (Figure 4c), *CSMuse* achieves an average win rate of 65.54% and an average improvement of $+0.0237$. Across individual domains, the win rates and average improvements are 48.98% and $+0.0201$ for Computing Methodologies, 64.00% and $+0.0252$ for Networks, and 83.67% and $+0.0258$ for Computer Systems Organization, respectively.

We further examine the LLM-as-a-Judge score to assess the quality gains attributable to cross-domain knowledge (Figure 4d). Overall, *CSMuse* is selected as the stronger approach in 46.62% of comparisons, compared with 16.89% for the primary-domain-only variant, and achieves an average LJ improvement of $+0.1194$. For Computing Methodologies, Networks, and Computer Systems Organization, *CSMuse* achieves win rates of 34.69%, 50.00%, and 60.00%, with average LJ improvements of $+0.1122$, $+0.1267$, and $+0.1190$, respectively. These results show that cross-domain knowledge transfer provides the largest benefits in the Networks and Computer Systems Organization domains, where research ideas often require complex architectural and system-level reasoning.

7 Conclusion

We present *CSMuse*, an evidence-grounded, cross-domain idea-generation system for complex and interdisciplinary computer science research. *CSMuse* integrates a scientific discovery dataset constructed from premier conferences, grounds generated ideas in evidence retrieved from domain-specific knowledge graphs, and employs cross-domain multi-agent collaboration to generate and refine research ideas. Experimental results demonstrate that *CSMuse* consistently produces high-quality ideas across different LLM backbones and research domains.

8 Statement

This research uses three publicly available datasets, ensuring compliance with data privacy policies. Our system is designed to augment, not replace, human researchers. The use of Large Language Models in this paper was restricted to polishing writing.

REFERENCES

ACM. Acm computing classification system. https://dl.acm.org/ccs, 2026. Accessed: 2026-09-26.

Dhruv Agarwal, Bodhisattwa Prasad Majumder, Reece Adamson, Megha Chakravorty, Satvika Reddy Gavireddy, Aditya Parashar, Harshit Surana, Bhavana Dalvi Mishra, Andrew McCallum, Ashish Sabharwal, et al. Autodiscovery: Open-ended scientific discovery via bayesian surprise. *arXiv preprint arXiv:2507.00310*, 2025.

Allen Institute for AI. Semantic scholar. https://www.semanticscholar.org/, 2026. Accessed: 2026-03-11.

Association for Computing Machinery. Acm digital library. https://dl.acm.org/, 2026. Accessed: 2026-03-11.

Jinheon Baek, Sujay Kumar Jauhar, Silviu Cucerzan, and Sung Ju Hwang. Researchagent: Iterative research idea generation over scientific literature with large language models. In *NAACL'25*, pp. 6709–6738. ACL, 2025.

Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S. Weld. Specter: Document-level representation learning using citation-informed transformers. In *ACL'20*. ACL, 2020.

Cornell University. arxiv.org e-print archive. https://arxiv.org/, 2026. Accessed: 2026-03-11.

DeepSeek. Deepseek. https://www.deepseek.com/, 2026. Accessed: 2026-09-26.

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

Alireza Ghafarollahi and Markus J Buehler. Sciagents: automating scientific discovery through bioinspired multi-agent intelligent graph reasoning. *Advanced Materials*, 37(22):2413523, 2025.

IEEE. Ieee xplore digital library. https://ieeexplore.ieee.org/, 2026. Accessed: 2026-03-11.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel S Weld. S2orc: The semantic scholar open research corpus. In *ACL'20*, pp. 4969–4983. ACL, 2020.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

John H Morris, Karthik Soman, Rabia E Akbas, Xiaoyuan Zhou, Brett Smith, Elaine C Meng, Conrad C Huang, Gabriel Cerono, Gundolf Schenk, Angela Rizk-Jackson, et al. The scalable precision medicine open knowledge engine (spoke): a massive knowledge graph of biomedical information. *Bioinformatics*, 39(2):btad080, 2023.

OpenAI. Chatgpt. https://chatgpt.com, 2026. Accessed: 2026-03-11.

OpenReview. Openreview. https://openreview.net/, 2026. Accessed: 2026-09-26.

Jason Priem, Heather Piwowar, and Richard Orr. Openalex: A fully-open index of scholarly works, authors, venues, institutions, and concepts. *arXiv preprint arXiv:2205.01833*, 2022.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Haoyang Su, Renqi Chen, Shixiang Tang, Zhenfei Yin, Xinzhe Zheng, Jinzhe Li, Biqing Qi, Qi Wu, Hui Li, Wanli Ouyang, Philip Torr, Bowen Zhou, and Nanqing Dong. Many heads are better than one: Improved scientific idea generation by a LLM-based multi-agent system. In *ACL'25*, pp. 28201–28240. ACL, 2025.

Don R Swanson. Undiscovered public knowledge. *The Library Quarterly*, 56(2):103–118, 1986.

Don R Swanson and Neil R Smalheiser. An interactive system for finding complementary literatures: a stimulus to scientific discovery. *Artificial intelligence*, 91(2):183–203, 1997.

Justin Sybrandt, Ilya Tyagin, Michael Shtutman, and Ilya Safro. Agatha: automatic graph mining and transformer based hypothesis generation approach. In *CIKM'20*, pp. 2757–2764. ACM, 2020.

USENIX Association. Usenix: The advanced computing systems association. `https://www.usenix.org/`, 2026. Accessed: 2026-03-11.

Bin Wang, Chao Xu, Xiaomeng Zhao, Linke Ouyang, Fan Wu, Zhiyuan Zhao, Rui Xu, Kaiwen Liu, Yuan Qu, Fukai Shang, et al. Mineru: An open-source solution for precise document content extraction. *arXiv preprint arXiv:2409.18839*, 2024a.

Qingyun Wang, Lifu Huang, Zhiying Jiang, Kevin Knight, Heng Ji, Mohit Bansal, and Yi Luan. Paperrobot: Incremental draft generation of scientific ideas. In *ACL'19*, pp. 1980–1991. ACL, 2019.

Qingyun Wang, Doug Downey, Heng Ji, and Tom Hope. Scimon: Scientific inspiration machines optimized for novelty. In *ACL'24*, pp. 279–299. ACL, 2024b.

A PROMPTS

A.1 PAPER SUMMARIZATION

The prompt used to generate a structured summary of each paper is shown in Figure 5.

A.2 KNOWLEDGE GRAPH QUERY

The prompt used to query the domain-specific knowledge graphs is shown in Figure 6.

A.3 SEED IDEA GENERATION

The prompt used to generate seed research ideas is shown in Figure 7.

A.4 MULTI-AGENT COLLABORATION

Idea Critic. The prompt used by the critic agent to evaluate seed ideas and provide refinement guidance is shown in Figure 9.

Domain-Specific Supplementation. The prompt used by domain agents to retrieve supporting evidence and supplement the idea is shown in Figure 8.

A.5 IDEA REFINEMENT

Cross-Domain Refinement. The prompt used to integrate insights and evidence from multiple domains is shown in Figure 10.

Novelty Refinement. The prompt used to assess and improve the novelty of the generated idea is shown in Figure 11.

B EXPERIMENTS

B.1 EXPERIMENT SETTINGS

Details of the papers collected from the selected premier conferences are provided in Table 3. The prompt used to generate the 150 research problems in the evaluation set is shown in Figure 12.

B.2 EXPERIMENT RESULTS

Detailed LLM-as-a-Judge scores for each baseline and ablation variant are reported in Tables 5, 4, 7, 6, 9, 8, and 10.

Table 3: Distribution of Papers Across Domains, Conferences, and Years

Research Domain	Conference	2021	2022	2023	2024	2025	2026
Computing Methodologies	ICLR	—	—	7	123	60	75
	ICML	—	—	60	75	147	37
	NeurIPS	—	—	127	121	136	—
Networks	ACM SIGCOMM	55	55	71	62	74	85
	USENIX NSDI	59	78	95	112	83	50
Computer Systems Organization	ACM SOSP	54	—	43	43	66	—
	USENIX OSDI	31	49	55	53	53	136

Prompt

You are a rigorous scientific paper information extraction assistant. Your task is to extract structured information from the input paper and output it strictly as JSON. You must rely only on the paper text. Do not guess or hallucinate. If a field is missing or unclear, use null, [], or "unknown".
Your goal is to compress the paper into a structured representation suitable for:
- idea discovery
- knowledge graph construction
- retrieval
- downstream evaluation

Focus on preserving:
- the core problem
- why it matters
- the key assumptions
- the central mechanisms
- the detailed format of each mechanism
- the system architecture
- inputs and outputs
- evaluation setup and results
- deployment constraints
- limitations and failure modes
- the most likely novelty signal
- related methods
- keywords

Extraction rules:
1. Use only information explicitly supported by the paper body, tables, figures, captions, and appendix.
2. Do not use outside knowledge to fill gaps.
3. Do not paraphrase into something the paper does not claim.
4. If multiple candidate answers exist, keep the most central and specific one.
5. For each mechanism, extract not only what it does, but also its operational format: input, process, output, trigger, constraints, and how it connects to other components.
6. For evaluation results, extract concrete numeric results whenever available. Include improvement percentages, accuracy, latency, throughput, overhead, ablation findings, or qualitative conclusions supported by experiments.
7. If a field is not explicitly stated, write null or [] rather than guessing.
8. Output must be valid JSON only. Do not include Markdown, explanations, or extra text.
9. Set paper_id to the supplied PAPER_ID exactly.
10. domain and subdomain are mandatory taxonomy labels, not free-form descriptions. domain must be a level-1 key. subdomain must be a level-3 descendant of that domain. The intervening level-2 label is only a path and must not be returned.

Field definitions:
- paper_id: use the PAPER_ID supplied by the caller exactly
- title: paper title
- domain: choose exactly one first-level key from the supplied topic taxonomy; copy its spelling exactly
- subdomain: choose exactly one third-level key under the selected domain (a child of any second-level key); copy its spelling exactly. Do not use a second-level key as subdomain
- problem: the core problem the paper addresses
- motivation: why the problem matters and what is lacking in prior work
- assumptions: key explicit or implicit assumptions the method depends on
- key_idea: the main novelty in one sentence
- mechanism: a list of concrete mechanism units. Each unit should describe one important technical mechanism with its detailed format
- system_architecture: pipeline stages or system components, in order
- evaluation.metrics: evaluation metrics
- evaluation.testbed: datasets, benchmarks, simulators, environments, or deployment settings
- evaluation.baselines: baselines or comparison methods
- evaluation.results: concrete experimental results and conclusions
- deployment_constraints: resource, environment, observability, latency, or engineering constraints
- limitations: stated or inferable limitations
- failure_modes: cases where the method may fail or behave badly
- novelty_clue: the most likely novelty signal, in short phrases
- related_methods: directly relevant methods, systems, or model names mentioned in the paper
- keywords: important keywords covering task, method, objects, and mechanisms

Mechanism extraction format: Each mechanism item must follow this structure: { "name": "short mechanism name", "purpose": "what this mechanism is used for", "process": ["ordered technical steps"], "trigger_or_condition": "when this mechanism is invoked, if stated", "constraints": ["important constraints, assumptions, or dependencies"], "connected_components": ["other components this mechanism interacts with"] }
Evaluation result extraction format: Each evaluation result item must follow this structure: { "claim": "what the experiment shows", "metric": "metric used", "value": "numeric value or qualitative result", "baseline": "baseline being compared against, if any", "setting": "dataset, benchmark, workload, topology, testbed, or experimental condition", "evidence_location": "section/table/figure name if available" }
Output format:
- Return only the following JSON object
- Keep all keys exactly as shown
- Do not add or remove top-level keys
- Use double quotes for strings
- Use strings inside arrays
- Use null or [] if no value is available
- Do not include comments, code fences, or any text outside the JSON

JSON template: { "paper_id": null, "title": null, "domain": null, "subdomain": null, "problem": null, "motivation": null, "assumptions": [], "key_idea": null, "mechanism": [{ "name": null, "purpose": null, "process": [], "trigger_or_condition": null, "constraints": [], "connected_components": [] }], "system_architecture": [], "evaluation": { "metrics": [], "testbed": [], "baselines": [], "results": [{ "claim": null, "metric": null, "value": null, "baseline": null, "setting": null, "evidence_location": null }] }, "deployment_constraints": [], "limitations": [], "failure_modes": [], "novelty_clue": [], "related_methods": [], "keywords": [] }
Paper ID supplied by the caller: {PAPER_ID}
Topic taxonomy (level 1 -> level 2 -> level 3): {TOPIC_TAXONOMY}
Now extract the information from the following paper text: {PAPER_TEXT}

Figure 5: Prompt used to generate structured summaries of the collected papers.

Find evidence relevant to this research problem and seed mechanism.
Problem: << PROBLEM >>
Seed: <<SEED_IDEA>>
Critic findings (empty means no critic was supplied): <<CRITIC>>
Prioritize concrete mechanisms that repair identified weaknesses, their operating assumptions, resource costs, limitations, and competing approaches. When findings are empty, use only the question and seed; do not assume a hidden critique.

Figure 6: Prompt used to query the domain-specific knowledge graphs.

Prompt

You are a scientific research ideation assistant.
Given a research problem and structured summaries of related papers from one domain-specific GraphRAG knowledge graph, propose novel, technically concrete research ideas. Use the paper summaries as evidence and inspiration, but do not copy a paper's contribution and do not claim that an idea has been experimentally validated.
Return valid JSON only. Do not use Markdown fences or any text outside the JSON. The root object must have an "ideas" array. Produce exactly {NUM_IDEAS} ideas. Each idea must have these keys:
idea_id: a stable string such as "idea_001"
title: concise idea title
claim: one-sentence central research claim
problem: the specific gap or limitation addressed
mechanism: concrete method and how it works
novelty: what is new compared with the supplied papers
expected_benefit: expected technical benefit
evaluation_plan: datasets/testbeds, baselines, and metrics
risks: important failure modes or assumptions
supporting_papers: paper_id strings from the supplied context, or []
The "supporting_papers" field may only contain paper IDs that appear in the supplied paper context. If no papers are supplied, return [].
Research problem: {PROBLEM}
Related paper summaries: {PAPER_CONTEXT}
Before finalizing, check that every idea is feasible, falsifiable, and distinct from the others. Keep the mechanism and evaluation plan specific enough to implement.

Figure 7: Prompt used to generate seed research ideas.

Prompt

You are the <<DOMAIN>> evidence specialist. Supplement the existing seed; do not propose an independent replacement idea. All domain specialists use this template.
Problem: << PROBLEM >>
Seed: <<SEED_IDEA>>
Critic findings (may be empty): <<CRITIC>>
Retrieved evidence: <<EVIDENCE>>
Return a compact Mechanism section containing at most two supported supplements. For each, give the supplied paper ID and supporting passage, addressed defect ID (or a question constraint when critic is empty), transferable operation, insertion point, prerequisites, resource/deployment cost, and a falsifiable ablation. Separate evidence from hypotheses. Do not invent paper IDs or measured benefits. Identify redundancy with the seed; do not count it as a new mechanism. If no evidence-backed supplement improves this seed within its constraints, output "Mechanism: No applicable supplement" and explain why. Do not force domain transfer.

Figure 8: Prompt used by domain agents to supplement ideas with domain-specific evidence.

Prompt

Evaluate the supplied seed against the question before any KG retrieval.
Problem: << PROBLEM >>
Seed: <<SEED_IDEA>>
Return the same assessment regardless of the later ablation condition:
Quality: problem fit, technical correctness, assumptions, feasibility, evaluation adequacy.
Prioritized defects: assign IDs D1, D2, ...; identify the affected operation, violated constraint or missing justification, severity, and a concrete repair objective.
Evidence needs: for each defect identify what mechanism, applicability condition, or prior-art comparison must be investigated. Do not invent papers or infer novelty from missing evidence. Do not prescribe which domain must supply the answer.
Revision guidance: preserve sound seed components and the original constraints; propose the smallest justified repairs and falsifiable tests. If no repair is needed, say so. Do not force new components, cross-domain fusion, or a replacement design. No retrieved evidence or secondary idea is available at this stage.

Figure 9: Prompt used by the critic agent to evaluate seed ideas and provide refinement guidance.

Prompt

Apply one revision pass. This exact template is shared across ablation conditions; only critic and evidence-backed supplements may differ. Do not infer missing inputs.
Problem: << PROBLEM >>
Primary domain: <<PRIMARY_DOMAIN>>
Shared seed: <<SEED_IDEA>>
Critic (empty means unavailable): <<CRITIC>>
Domain supplements (empty object means no retrieval): <<SUPPLEMENTS>>
Preserve the problem constraints and sound seed mechanisms. Repair concrete defects with the smallest defensible changes. Assess supplied supplements for correctness, applicability, redundancy and added cost. Adopt or simplify only when they address an actual weakness; reject unsupported or unnecessary additions. Domain labels are metadata, not evidence of benefit. Never invent missing retrieved evidence. If no secondary mechanism is useful, use no cross-domain fusion: the final mechanism and revision guidance must contain only seed-focused repairs. Keep rejected options only in the decision audit, not in the final design or required experiments.
Hard-constraint gate (identical for every revision condition):
Before revising, extract the explicit requirements from the Question: SLO values, latency percentiles and measurement windows, throughput, safety/correctness, precision/accuracy, workload, resource budget, and prohibited modifications. Preserve their original scope and numbers. Distinguish mandatory requirements from preferences; do not invent constraints or treat seed/critic assumptions as permission to override the Question. If ambiguous, state the ambiguity.
Check every proposed repair and supplied supplement, then the complete combined mechanism, against those requirements. Account for interactions and total costs, including queueing, admission/rejection, retries, transfer, and control overhead.
Never claim a repair by relaxing the SLO, substituting average for tail latency, narrowing the tested workload, excluding failures from the denominator, adding resources, weakening safety, or reducing precision/accuracy beyond what the Question explicitly permits. Approximation, quantization, rerouting, skipping, and modified hardware/protocol assumptions require explicit compatibility.
In particular, a least-unsafe action, a degraded-SLO contract, or best-effort execution is not compliance with a mandatory safety/SLO requirement. Queueing, rejection, or stopping may be safer but must still be checked against the original availability, latency and throughput requirements; do not present them as a free solution. Describe out-of-contract emergency behavior only as a limitation.
Reject an incompatible supplement and retain a compliant seed operation when one exists. If the seed itself is noncompliant, repair it without changing the requirements. If no defensible repair is available, mark the requirement unresolved or violated and state that the claimed target is not established. Do not fabricate feasibility or experimental confirmation.
Report a compact constraint audit in What changed: original requirement, affected operation, status (supported by design / unverified / violated), and justification plus a falsifying test where needed. Design support is not measured compliance. Use the same audit and output budget even when no supplements exist.
Use the same detail budget in every condition, aiming for 1200-1600 words total:
Revised idea: Title, Claim, Mechanism (explicit operations and assumptions).
What changed: defect addressed and why the change is necessary.
Evaluation plan: relevant baselines, measurable metrics, main-mechanism ablation, stress/falsification case, and resource overhead. For adopted supplements include a with/without ablation holding other changes fixed. These are plans, not results.
Remaining risks and unresolved evidence needs.
Fusion decisions: end with a fenced JSON object {"fusion_decisions": [...]}. Record secondary-domain mechanisms only. Each entry has source_domain, mechanism, decision (adopt/simplify/reject), already_in_seed (boolean), reason, final_evidence. For adopt/simplify quote the exact integrated operation in the final Mechanism; for reject use an empty final_evidence. If no secondary supplement was supplied, use an empty array. Do not invent fusion percentages.

Figure 10: Prompt used for cross-domain idea refinement.

Prompt

Revise the supplied full proposal only to reduce substantive overlap with the pre-2026 evidence. Treat this as a controlled ablation: preserve the seed, critic decisions, cross-domain fusion, and output detail unless a repeated mechanism is genuinely unnecessary.
Problem: <<PROBLEM>>
Full proposal: <<FULL_IDEA>>
Pre-2026 overlap evidence (may be empty): <<HISTORICAL_EVIDENCE>>
Identify repeated mechanisms that are non-essential. Modify only those parts, prefer adding a distinct decision rule, trigger, feedback loop, or composition boundary over deleting useful operations. Every change must preserve the original SLO, safety/correctness, precision/accuracy, workload, and resource requirements. Never improve novelty by relaxing a requirement, narrowing the workload, excluding failures, or weakening safety or precision. Retain all components required to satisfy those requirements. Do not claim that lexical rewriting is novelty. If no defensible non-essential overlap exists, return the full proposal unchanged and state that no repair was made.
Output the same format as the full proposal, including Title, Claim, Mechanism, What changed, Evaluation plan, Remaining risks, and a compact constraint audit. End with a fenced JSON object {"novelty_repairs": [...]} listing each changed operation, its overlap evidence, why it was non-essential, and the preserved requirement.

Figure 11: Prompt used for novelty-oriented idea refinement.

Prompt

You are extracting research problems from structured paper summaries.
Read exactly one paper summary below and identify the single central problem the paper is trying to solve. Return one concrete, self-contained question. Keep the problem faithful to the summary: include the setting, the central technical or operational difficulty, and important constraints when explicitly available. Do not describe the paper's solution, method, or results and do not invent requirements. This problem will be used for downstream idea generation.
Return valid JSON only, with exactly these keys: {"paper_id": "...", "title": "...", "problem": "How can ...?"}
The problem must be one question sentence beginning with "How can". Extract exactly one problem for this paper, even if the summary lists several subproblems.
Paper summary: {SUMMARY}

Figure 12: Prompt used to generate the research problems in the evaluation set.

Table 4: AI-Scientist Evaluation Across Different Datasets Using *GPT-5.5*

Dataset	Problem Fit	Mechanism Specificity	Technical Soundness	Evaluation Quality	Novelty	Feasibility
CM	4.2067	3.4800	2.5867	3.3467	3.0400	2.9200
Net	4.3197	3.5782	2.6667	3.3197	2.9524	2.6599
CSO	4.2313	3.5646	2.4150	3.3401	3.0136	2.4762
All	4.2523	3.5405	2.5563	3.3356	3.0023	2.6869

Table 5: AI-Scientist Evaluation Across Different Datasets Using *DeepSeek-V4-Pro*

Dataset	Problem Fit	Mechanism Specificity	Technical Soundness	Evaluation Quality	Novelty	Feasibility
CM	4.5867	3.8400	3.2667	4.2067	3.5133	3.3000
Net	4.6467	3.9867	3.3600	4.2933	3.5267	3.3067
CSO	4.5800	4.0600	3.3867	4.3067	3.5533	3.2600
All	4.6044	3.9622	3.3378	4.2689	3.5311	3.2889

Table 6: VIRSCI Evaluation Across Different Datasets Using *GPT-5.5*

Dataset	Problem Fit	Mechanism Specificity	Technical Soundness	Evaluation Quality	Novelty	Feasibility
CM	3.9600	3.3000	2.3800	3.0467	2.8267	2.7933
Net	4.1800	3.5067	2.6000	3.2267	2.9467	2.6533
CSO	4.0267	3.3200	2.3067	3.0867	2.7733	2.4733
All	4.0556	3.3756	2.4289	3.1200	2.8489	2.6400

Table 7: VIRSCI Evaluation Across Different Datasets Using *DeepSeek-V4-Pro*

Dataset	Problem Fit	Mechanism Specificity	Technical Soundness	Evaluation Quality	Novelty	Feasibility
CM	4.6733	3.8467	3.2800	4.2200	3.6133	3.3600
Net	4.5800	3.9133	3.3333	4.2067	3.4667	3.3400
CSO	4.5267	4.2600	3.4533	4.4400	3.5467	3.2533
All	4.5933	4.0067	3.3556	4.2889	3.5422	3.3178

Table 8: Idea Generation Performance Across Methods in Networks

Method	Problem Fit	Mechanism Specificity	Technical Soundness	Evaluation Quality	Novelty	Feasibility
Vanilla LLM	3.6200	2.7600	2.5200	3.1000	2.5600	3.1000
LLM with Retrieval	4.6000	4.4400	3.8600	4.6200	3.2200	3.6800
LLM with Refinement	4.6800	4.5600	4.2000	4.8400	3.1000	3.6400
LLM with Retrieval and Refinement	4.7800	4.5800	4.2800	4.8000	3.2200	3.6800
CSMuse	4.8200	4.9200	4.3400	4.9400	3.5400	3.5400

Table 9: Idea Generation Performance Across Methods in Computing Methodologies

Method	Problem Fit	Mechanism Specificity	Technical Soundness	Evaluation Quality	Novelty	Feasibility
Vanilla LLM	3.6800	2.7600	2.4400	3.0800	2.5600	2.9800
LLM with Retrieval	4.6000	4.4200	3.9000	4.6400	3.3800	3.3000
LLM with Refinement	4.7800	4.7200	4.3600	4.8200	3.3000	3.5400
LLM with Retrieval and Refinement	4.7600	4.7200	4.3600	4.8600	3.2800	3.5200
CSMuse	4.8800	4.9000	4.4800	4.9400	3.5200	3.4600

Table 10: Idea Generation Performance Across Methods in Computer Systems Organization

Method	Problem Fit	Mechanism Specificity	Technical Soundness	Evaluation Quality	Novelty	Feasibility
Vanilla LLM	3.5200	2.6800	2.3600	2.9200	2.5600	2.9000
LLM with Retrieval	4.5000	4.4200	3.8400	4.4400	3.1200	3.2200
LLM with Refinement	4.6000	4.5600	4.2000	4.7400	3.0800	3.4200
LLM with Retrieval and Refinement	4.6000	4.6200	4.1400	4.6600	3.1400	3.3600
CSMuse	4.6400	4.8200	4.3000	4.8800	3.4000	3.2600