A Multi-Agent System for Detection and Prioritization of Research Gaps in Scientific Literature

Rodrigo Silva[1], **Bruno Lemos**[1], **Bruna Falcucci**[1], **Guilherme Maia**[1]

[1]Department of Computer Science – Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brazil

{rodrigo.silva, bruno.lemos, bruna.falcucci, jgmm}@dcc.ufmg.br

Abstract. *Identifying promising research gaps remains a significant challenge in scientific research. As the volume of published literature increases, determining which questions remain unresolved and which approaches have been previously explored becomes more complex. This study introduces a multi-agent system designed to detect and prioritize candidate research gaps from either a research theme or a corpus of scientific papers. The system systematically maps the literature, identifies evidence-based research opportunities, evaluates their feasibility, and generates a prioritized report to support researchers during the literature review process. Instead of relying on a single language model, the workflow is divided into specialized components with clearly defined responsibilities. Deterministic modules handle tasks such as resource retrieval and ranking, while large language model (LLM)-based agents handle tasks requiring semantic interpretation, including evidence synthesis, classification, and critique. Candidate gaps are identified from two independent sources of evidence: infrequent but meaningful associations between scientific concepts extracted from an entity co-occurrence graph, and explicit statements of limitations or future work reported by original authors. Each candidate is linked to the corresponding supporting passages, assigned a confidence level, and excluded if the available evidence is insufficient. Experiments conducted on three literature corpora demonstrate that 90% of the reported candidates are supported by verbatim evidence extracted from the source documents. In contrast, a single-prompt baseline produces no evidence-grounded suggestions and generates more generic or trivial recommendations. Although the proposed system identifies fewer candidates, the resulting reports offer higher precision and greater traceability.*

1. Introduction

Identifying meaningful research gaps requires reviewing extensive literature, understanding prior work, and determining which unresolved questions remain relevant and feasible. Gaps may involve underexplored concept combinations, missing experiments, unsupported assertions, or methodological limitations. As scientific literature grows across increasingly diverse venues, performing this analysis manually becomes impractical [Abd-alrazaq et al. 2024, Salem et al. 2025].

Large language models (LLMs) support tasks such as literature review, scientific question answering, manuscript assessment, and idea generation [Wang et al. 2024, Lála et al. 2023, Liang et al. 2024, Si et al. 2024]. However, research-gap identification

remains challenging: single-prompt methods often produce generic suggestions, blur distinct evidence types, and lack traceability to source documents, reducing their reliability for research workflows.

To address these challenges, we introduce a multi-agent system that detects and prioritizes candidate research gaps. Deterministic modules handle retrieval and ranking, while LLM-based agents perform semantic tasks such as synthesis, classification, and critique. Gaps are detected from rare but meaningful entity associations in a co-occurrence graph and from explicit limitations or future-work statements. Each candidate is linked to supporting evidence, assigned a confidence level, and included in a prioritized report.

The proposed system is designed to support researchers rather than replace them. It does not purport to identify definitive research gaps or to make autonomous decisions. Instead, it generates evidence-supported candidates for user inspection and validation. When available evidence is insufficient, the system explicitly refrains from issuing a recommendation.

2. Background and Related Work

2.1. LLM-assisted literature work

LLMs increasingly support scientific literature analysis. AutoSurvey generates survey drafts through retrieval and refinement [Wang et al. 2024], LitLLM decomposes related-work writing into retrieval and generation [Agarwal et al. 2024], and PaperQA and OpenScholar provide citation-backed scientific question answering [Lála et al. 2023, Asai et al. 2024]. Unlike these systems, our focus is specifically on detecting and ranking candidate research gaps with traceable evidence.

2.2. Research gap detection

Closest to our goal, a few systems explicitly target research gaps. A topic-modeling approach identifies gaps in COVID-19 literature [Abd-alrazaq et al. 2024] as under-populated or weakly connected clusters, but provides neither evidence for individual suggestions nor feasibility assessment. GAPMAP [Salem et al. 2025] uses LLMs to map explicit and implicit biomedical gaps at scale, but does not prioritize what to pursue next. Both are domain-specific, single-technique approaches without feasibility judgments or auditable prioritization. More broadly, our rarity signal follows Swanson's literature-based discovery [Swanson 1986], treating hidden connections between disjoint literatures as discoverable knowledge.

Language models have been adapted to scientific text through domain pretraining (SciBERT [Beltagy et al. 2019]), science-focused generation (Galactica [Taylor et al. 2022]), and claim verification [Wadden et al. 2020]. LLMs have also been used for manuscript feedback, with targeted prompts outperforming generic ones [Liu and Shah 2023] and GPT-4 showing overlap with human reviews [Liang et al. 2024]. More autonomous systems such as the AI Scientist [Lu et al. 2024] and studies of LLM-generated research ideas [Si et al. 2024] further motivate grounded, human-supervised designs that avoid over-claiming autonomy.

2.3. Multi-agent systems and LLM agents

Decomposing complex tasks across cooperating agents is well established [Wooldridge 2009]. Recent frameworks such as AutoGen [Wu et al. 2023],

MetaGPT [Hong et al. 2024], and CAMEL [Li et al. 2023] coordinate role-specialized LLM agents, while ReAct [Yao et al. 2023], Reflexion [Shinn et al. 2023], and Self-Refine [Madaan et al. 2023] structure reasoning and self-correction. ResearchAgent similarly uses collaborating agents to refine research ideas [Baek et al. 2025]. Our system instead uses Chain-of-Thought [Wei et al. 2022] with schema-validated outputs for grounded single-pass tasks, while keeping resource lookup and control flow deterministic.

2.4. Positioning within agent and MAS research

From an agent-theoretic perspective, the system is a constrained multi-agent organization defined by four elements. *Roles*: five agents have distinct responsibilities. *Environment*: they coordinate through a persisted typed blackboard. *Protocol*: LLM outputs are schema-validated, and the Reviewer critiques the Writer. *Control*: sequencing remains under a deterministic state machine. This contrasts with AutoGen [Wu et al. 2023], MetaGPT [Hong et al. 2024], and CAMEL [Li et al. 2023], where dialogue coordinates agents. For grounded document analysis, we treat limited agent autonomy as an explicit, measurable design choice (Sections 3.1 and 4).

Existing tools each cover part of the space-review synthesis, question answering, manuscript feedback, or single-technique gap detection-but none combine multi-source gap detection, feasibility-aware prioritization, a multi-agent architecture, an inspection checkpoint before the expensive stage, and a structured report in which each suggestion is linked to its source evidence. Our contribution is the integration of these elements around the specific problem of suggesting and ranking candidate gaps.

3. Methods

The system transforms a free-text theme or existing corpus into a ranked, evidence-backed list of candidate research gaps. Users may provide a theme, an explicit query, or a pre-ingested corpus. The workflow comprises six stages-ingestion, mapping, gap detection, feasibility assessment, prioritization, and explanation-coordinated by a state-machine orchestrator. An optional human-in-the-loop checkpoint pauses execution after gap detection for candidate review before feasibility assessment. Figure 1 summarizes this flow.

The stage boundaries are concrete. *Ingestion* builds a deduplicated corpus of titles, abstracts, and metadata from arXiv, OpenAlex, Semantic Scholar, and the ACL Anthology. *Mapping* produces embeddings, topics, typed entities, and a co-occurrence graph using `all-MiniLM-L6-v2`, BERTopic, and entity dictionaries. *Gap detection* generates typed candidates with evidence quotes through phrase patterns and an LLM classifier. *Feasibility* assigns six scores, tier labels, and an attackable verdict using external resource searches and an LLM classifier. *Prioritization* ranks candidates deterministically, and *explanation* assembles the auditable report.

3.1. Why decomposition, and why Chain-of-Thought

Asking one prompt to "find the gaps" collapses retrieval, structure extraction, evidence classification, feasibility assessment, and ranking into a single opaque step. Decomposition assigns each task to an appropriate mechanism, improves auditability, and localizes failures. Searches and computations, such as resource lookup and ranking, are

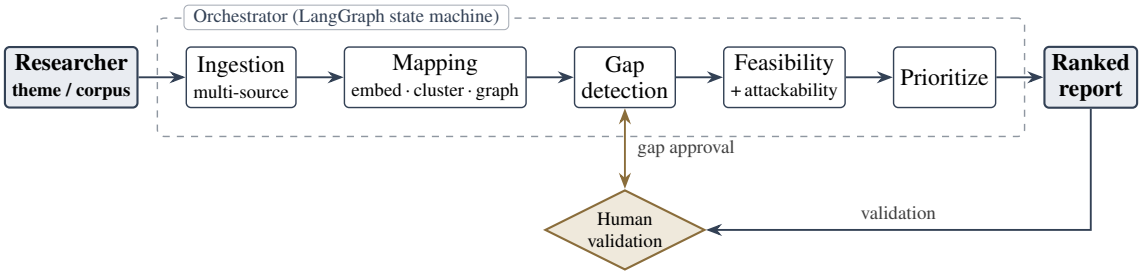

Figure 1. System overview: a theme or corpus flows through the six-stage pipeline to a ranked, evidence-backed report.

handled deterministically for reproducibility and reliability. LLM agents instead use Chain-of-Thought [Wei et al. 2022] with schema-validated outputs rather than open-ended ReAct [Yao et al. 2023] loops, since the sub-tasks are single-pass grounded extraction and judgment problems that benefit more from structured reasoning than dynamic tool selection.

Our agents are synchronous and follow a fixed schedule: they do not run concurrently, re-plan, or negotiate. This is a deliberate choice for grounded single-pass extraction, where dynamic behavior would add cost and variance with little expected benefit. The fixed sequence improves reproducibility, localizes failures, and enables component-level ablations (Section 4). Its main limitation is reduced adaptability: the system cannot react to partial results, revisit earlier stages, or exploit concurrency, making it unsuitable for tasks such as live literature monitoring.

3.2. Agents and responsibilities

Five agents and an orchestrator compose the system; Table 1 summarizes their inputs, roles, and outputs. Deterministic agents provide auditable traces: the **Researcher** logs dataset, benchmark, and framework checks, while the **Prioritizer** applies a fixed scoring function. LLM agents support self-correction: the **Writer** produces grounded gap summaries from supplied evidence, and the **Reviewer** critiques and either approves or requests one rewrite. The **Query Planner** derives queries and sources from a theme, while the **Orchestrator** sequences stages, manages shared state, persists progress, and retains deterministic control flow.

3.3. Gap detection

Gap detection combines two complementary signals over the mapped corpus, which the mapping stage has embedded, clustered into topics, and turned into typed entities (methods, datasets, tasks, metrics) and an entity co-occurrence graph [Reimers and Gurevych 2019, Grootendorst 2022]. The first signal looks for rare-but-plausible entity pairs: two entities each individually common in the corpus but rarely appearing together, whose types form an allowed combination (e.g. method–task or method–dataset). For a pair (a, b), we use normalized pointwise mutual information [Bouma 2009] over document co-occurrence statistics (Equation 1).

$$\mathrm{npmi}(a,b) = \frac{\ln \frac{p(a,b)}{p(a)\,p(b)}}{-\ln p(a,b)} \in [-1,1], \qquad r(a,b) = \frac{1-\mathrm{npmi}(a,b)}{2} \in [0,1], \tag{1}$$

Table 1. Agent responsibilities (D = deterministic, L = LLM-based).

Agent	Input	Processing role	Output	Kind
Query Planner	Theme	Derive a focused search query and choose sources	Query, sources, year range	L
Researcher	Gap (entities, terms)	Dataset, benchmark and framework availability checks	Per-resource scores, matches, trace	D
Writer	Entities + evidence quotes	Draft grounded gap summary / feasibility justification	Summary, rationale	L
Reviewer	Draft + source context	Critique; approve or request one rewrite	Verdict, critique, edits	L
Prioritizer	Gaps + assessments	Weighted ranking by feasibility, rarity, support	Ranked gaps + rationale	D
Orchestrator	Run config	Sequence stages, merge outputs, build report	Auditable report	D

Here $p(\cdot)$ are corpus frequencies and $r(a, b)$ the rarity score (higher is rarer), operationalizing literature-based discovery [Swanson 1986] as a graph statistic over typed entities, analogous to bibliometric co-occurrence mapping [van Eck and Waltman 2010].

The second signal identifies sentences expressing future work, limitations, under-explored topics, or open questions using phrase patterns and an LLM classifier that also returns supporting spans. Both signals operate only on titles and abstracts, since current connectors do not provide the full text required by the section-aware extraction path. Candidates are typed as rare combinations, limitations, future work, or under-explored topics, preserving these evidence categories separately.

Every candidate is provided with the documents and quotes that support it. The Writer summarizes the candidate from at most a few of these quotes, and the Reviewer checks the draft against them; prompt inputs are sanitized to mitigate injection, and resource names produced by the model are filtered to those that were actually supplied, so the system cannot invent datasets or baselines.

3.4. Feasibility and attackability

For each candidate gap, six dimensions are scored on the $[0, 1]$ scale. The deterministic *dataset*, *benchmark*, and *framework* dimensions measure resource availability from extracted entities and external API lookups. The LLM-based dimensions are classified as *low*, *medium*, or *high*, mapped to $\{1.0, 0.5, 0.1\}$: *cost* estimates compute requirements, *complexity* implementation effort, and *maturity* how unproven the required techniques are (*low* = well established). In each case, lower difficulty or risk receives a higher score. Overall feasibility is the normalized weighted mean in Equation 2, with uniform default weights ($1/6$ each) that can be overridden per request.

$$overall = \sum_k w_k \, s_k \tag{2}$$

A deterministic rule then yields an explicit *attackable* verdict, true if *all* of the following hold: (i) *overall* $\geq \tau$ with default threshold $\tau = 0.5$; (ii) a dataset or a benchmark is available; (iii) the estimated complexity tier is not *high*; and (iv) the gap carries at least one method, dataset, or task entity. The verdict and its step-by-step rationale are authoritative and reproducible; an optional LLM pass adds a natural-language explanation grounded only in the supplied scores and matched resources, and never overrides the verdict.

3.5. Prioritization

The Prioritizer ranks gaps with a transparent weighted score as shown in Equation 3.

$$s(g) = w_f\, f(g) + w_r\, r(g) + w_s \cdot \frac{\min(\mathrm{support}(g),\, C)}{C}, \tag{3}$$

where f is overall feasibility, r is rarity, $\mathrm{support}$ is the number of corpus documents jointly supporting the gap (capped at C=5), and the weights default to w_f=0.55, w_r=0.35, w_s=0.10, and are renormalized to sum to one. Putting feasibility first reflects the assistant's goal: to surface opportunities that a researcher could realistically act on. Because the weighting is explicit, individual terms can be set to zero to perform ablations (Section 4).

3.6. Auditable report, confidence, and abstention

Each reported gap receives a confidence level based on corpus support n and evidence strength σ. Here, n counts supporting documents that yield at least one evidence span, while σ is the mean classifier confidence over the m extracted quotes. These quantities capture complementary evidence signals and are combined by Equation 4.

$$\mathrm{confidence} = \begin{cases} \textit{abstain} & \text{if } n = 0 \text{ and } m = 0, \\ \textit{high} & \text{if } n \geq 2 \text{ and } \sigma \geq 0.66, \\ \textit{medium} & \text{if } n \geq 1 \text{ and } (\sigma \geq 0.33 \text{ or } m \geq 1), \\ \textit{low} & \text{otherwise.} \end{cases} \tag{4}$$

When there is neither corpus support nor any extracted evidence, the system abstains rather than emit an unsupported recommendation. The final report keeps observed facts separate from inferred opportunities and records, for every gap, its supporting quotes, feasibility verdict, and confidence label, so that the entire output is auditable.

3.7. Orchestration, coordination, and the human in the loop

The orchestrator is a state machine whose nodes are the pipeline stages: shared state is threaded between nodes, each stage appends its summary to a merged statistics object, and sub-job identifiers are published live so progress can be followed.

When enabled, the checkpoint raises a *dynamic* interrupt after gap detection, persists the candidates and run state, and waits for a decision through the HTTP API. Resuming re-executes only the approval node, avoiding repeated detection or premature feasibility work. In the current prototype, the checkpoint acts as an auditable gate rather than a filter: it records the decision but does not yet modify the candidate set passed to feasibility.

3.8. Implementation

The prototype is implemented as a Python service with asynchronous execution and LangGraph-based orchestration [LangChain, Inc. 2024]. The provider-agnostic LLM gateway uses JSON-mode responses and content-hash caching for reproducibility. Major components are abstracted behind interfaces for testing, and the implementation includes 344 automated tests with 93% line coverage. We will release the source code and reproducibility artifacts upon acceptance.

4. Results

We walk through a real run on the theme "graph neural networks for drug discovery". The system ingests 119 papers from OpenAlex, maps the research space, and detects six candidate gaps; Table 2 shows the two gaps the report labeled HIGH confidence, exactly as it renders them.

The same run is also exposed through a lightweight web interface that lets a researcher start an analysis, browse the ranked gap list, and open each opportunity card to inspect its supporting evidence, confidence label, and feasibility cues.

Table 2. Two high-confidence opportunities from a real run on GNNs for drug discovery (119 papers); these are the two candidates the report labeled HIGH, at ranks 2 and 4 of six. Bars: rarity and feasibility in $[0, 1]$; "Conf.": confidence label.

Rank	Candidate	Supporting evidence (quote)	Rarity	Feas.	Conf.
2	Transformer × Drug Discovery *future work*	"… broader industry adoption of AI-driven PK."	0.54	0.38	HIGH
4	Drug Discovery × GAT *underexplored*	"different ML algorithms offer diverse VS profiles."	0.48	0.38	HIGH

The example illustrates the design. Both candidates are typed and grounded in a verbatim corpus quote the researcher can verify, and each carries a confidence label. The feasibility stage marks both not yet attackable-well-evidenced but of high estimated complexity and matching no standardized benchmark-so the report separates "well-evidenced" from "ready to act on now", a distinction a single "find the gaps" prompt does not surface. On another theme, the rarity filter produced a single weakly supported candidate, and the report abstained rather than asserting it, exercising the built-in hedge against over-claiming.

We evaluate the system on three real literature corpora against a single-prompt baseline using `gpt-4o-mini` as an automatic judge. The results are subject to three caveats: the judge is not a panel of human experts and is the same model used to generate candidates for both approaches; the system is deliberately selective; and the two arms are not fully matched. We therefore report both absolute counts and proportions and make these confounds explicit.

For each theme, we ingest an OpenAlex corpus and run both the full pipeline and a single-prompt baseline. The baseline returns JSON candidate gaps with a brief rationale grounded in the abstracts. An LLM judge scores relevance on a 1–5 scale and flags trivial suggestions; "useful" denotes relevance ≥ 4. Grounding is measured as the fraction of gaps linked to at least one verbatim corpus quote. Generation and judging use `gpt-4o-mini` with response caching; semantic deduplication removed no system candidates in any run. Table 3 summarizes the corpora and resulting gaps.

Three asymmetries limit this comparison. First, the baseline does not return verbatim spans, so its 0% grounding follows from the output contract rather than direct measurement. Second, it reads only the first 40 documents with abstracts truncated to 600 characters, whereas the system processes all abstracts in full. Third, baseline outputs are not deduplicated, while system candidates are, making raw candidate counts only partially comparable.

Table 3. Evaluation corpora (OpenAlex) and gaps produced by each approach.

Theme	Papers	System gaps	Baseline gaps
Graph neural networks for drug discovery	119	6	19
LLM agents for software engineering	118	3	19
Reinforcement learning for robotic manipulation	118	1	19
Total	**355**	**10**	**57**

The harness compares the full ranking against retrieval-only, no-feasibility, no-prioritization, and single-agent variants using rank correlation and standard ranking metrics, based on the judge's relevance scores. We additionally report grounding, usefulness, triviality, and mean relevance; Table 4 summarizes these measures.

Table 4 pools the comparison over the three corpora, and the two approaches behave very differently. The baseline is prolific (57 gaps vs. 10), but none are grounded in a verbatim quote; only 7% are judged useful, and 10.5% trivial. The system is far more selective: 90% of its gaps are grounded, 40% useful, and none trivial, with the single ungrounded gap automatically ABSTAINED rather than asserted. Across its gaps the report assigned 2 HIGH, 7 MEDIUM, and 1 ABSTAIN label.

Two cautions apply to reading this table. Grounding reflects an architectural difference more than an empirical one: the system emits spans because its output contract is built around them, and the baseline cannot, because its contract has nowhere to put them. Usefulness is not independent of that difference either, since the judge is instructed to penalize candidates that lack supporting evidence and receives the literal marker "no extracted evidence" for every baseline gap. The two measures are best read as two views of one design decision rather than as two independent findings.

Both approaches yield four useful gaps, but this pooled tie does not imply agreement. Per corpus, the system scores 2/6, 2/3, and 0/1 useful gaps, while the baseline scores 2/19, 0/19, and 2/19, so the useful sets differ across themes. We did not measure item-level overlap, which remains future work. With roughly one-sixth as many candidates, the system instead emphasizes evidence-backed precision and auditability at the cost of recall.

Table 4. Multi-agent system vs. single-prompt baseline, pooled over the three corpora.

Method	Gaps	Grounded	Useful	Trivial	Rel.
Multi-agent system	10	**90%**	**40%**	**0%**	3.4
Single-prompt baseline	57	0%	7%	10.5%	3.0

The ablations (Table 5, on the graph-neural-networks corpus) produce substantially different rankings. Relative to the full system, retrieval-only diverges most (Spearman $\rho = 0.21$), followed by single-agent (0.60), no-feasibility (0.89), and no-prioritization (0.94). However, divergence does not indicate ranking quality. Against the judge's relevance labels, retrieval-only matches the ordering exactly ($\rho = 1.00$), followed by single-agent (0.83, $p = 0.04$), no-feasibility (0.62), no-prioritization (0.41), and the full system (0.21, $p = 0.69$). Although six gaps from one corpus are insufficient for strong conclusions, the results do not support our weighting. As discussed in Section 5, the support term is also nearly inert in this corpus. All strategies re-rank the same candidate set, so these ablations affect ranking rather than detection.

Each theme took three to four minutes. Token accounting is instrumented only on the gap-detection gateway, so we report a lower bound rather than a total: \$0.0087–\$0.0116 per theme, the upper figure coming from the one run that executed with a cold response cache. Mapping, feasibility, and the judging passes are not yet metered.

Table 5. Ablation: rank divergence from the full system (Spearman ρ; GNN corpus).

Strategy	Spearman ρ vs. full system
retrieval-only (raw support)	0.21
single-agent (one-prompt ranking)	0.60
no-feasibility (ablation)	0.89
no-prioritization (rarity order)	0.94

4.1. Qualitative analysis

Qualitatively, the grounded gaps are concrete, typed, and linked to corpus evidence, while the attackability verdict distinguishes "well-evidenced" from "ready to act on now". The baseline instead produces fluent but generic, unverifiable, and sometimes trivial or near-duplicate suggestions. Such outputs consume researcher attention without improving actionability and can inflate apparent recall. The system's main weakness is the opposite: its conservative rarity filter reduces recall, producing only one abstained candidate on one theme.

5. Discussion

The system is most useful as a *triage and explanation* tool: it narrows a large body of literature to a ranked shortlist and shows the evidence behind each item, which is valuable for researchers and especially for graduate students entering a field. It is least useful as an oracle; it does not, and is not meant to, certify that a suggestion is a true, novel gap.

Hallucination is the central risk in any LLM-assisted research tool, and we mitigate it structurally rather than by trusting the model: suggestions are grounded in extracted quotes, model-produced resource names are filtered to those actually supplied, the report separates observed facts from inferred opportunities, the system abstains under weak evidence, and the optional checkpoint blocks the downstream stages until a researcher has inspected the candidates and recorded a decision. More broadly, the deterministic/LLM split is what lets us claim transparency-the parts a reviewer would most want to trust (resource lookups, ranking, the attackable verdict) are computations, not generations.

"Research gap" is itself contested; our proxies capture some kinds of gaps and miss others, such as deep methodological flaws, and the evaluation dimensions are operational stand-ins for an ultimately subjective notion of usefulness. The system-gap sample is small (ten gaps across three corpora), so the proportions are coarse and we rely on robust counts rather than p-values. The grounding contrast (90% vs. 0%) needs no judge, but as noted above it measures a difference in output contracts rather than in reasoning ability, and is not the stronger empirical claim.

The candidate filter is highly restrictive: pairs are considered only when both entities appear in at least three documents, co-occur in at most two, and form an allowed method–task or method–dataset combination. This ceiling limits the support term in Equation 3 to 0.4 of its range, so at weight 0.10 it changes the final score by at most 0.04, explaining the small effect of removing prioritization. Entity extraction is also sparse, with only 51, 29, and 33 documents contributing co-occurrence edges. Confidence has additional limitations: σ reflects whether an abstract states a gap rather than whether the entity pair constitutes one, pattern-extracted quotes count in m without contributing to σ, and n includes only supporting documents that yielded an evidence span.

The system performs no temporal reasoning: it uses publication year only during ingestion, and gaps carry no temporal information. Consequently, a gap reported in older work may still receive HIGH confidence even if later studies have already addressed it, since abstention is triggered by missing rather than superseded evidence. Detecting whether subsequent literature has closed a candidate is therefore a key extension of the current design.

The evaluator is an LLM rather than a panel of human experts, so its scores may be miscalibrated or biased toward LLM-written text; moreover, the same model generated both arms, although the implementation supports a separate judge. The maturity dimension provided no discrimination, assigning *medium* to all nine assessed gaps. Detection is limited to titles and abstracts, and the entity dictionaries and allowed type combinations are CS/ML-centric. Generalization to other domains, non-English text, larger corpora, and other LLM providers remains untested.

6. Conclusion and Future Work

We presented a multi-agent system that suggests and prioritizes candidate research gaps to support human researchers. The contribution is architectural, and it is aimed at the agents community as much as at literature tooling: it is a concrete, measured placement of the boundary between deterministic and LLM-based work inside a multi-agent document-analysis system. Role-specialized Chain-of-Thought agents communicate through a typed shared state under a fixed coordination protocol; every suggestion is grounded in corpus

evidence, feasibility is assessed, and an auditable, ranked report abstains when evidence is weak and can be gated on a recorded human decision before its expensive stage. On three real corpora, the system's gaps are far more often grounded and judged useful, and never trivial, than a single-prompt baseline's; its main limitation is the mirror image of that selectivity-recall. Future work therefore centers on less conservative candidate generation, temporal verification of whether later work has already closed a candidate, mining full text rather than abstracts alone, a human-expert study to complement the LLM judge, matched baselines with item-level overlap analysis, broader domains and providers, and a public reproducibility package. We believe the architecture is a sound basis for these steps and a useful template for agent-based research-assistance tools.

References

Abd-alrazaq, A., Nashwan, A. J., Shah, Z., et al. (2024). Machine learning–based approach for identifying research gaps: COVID-19 as a case study. *JMIR Formative Research*, 8:e49411.

Agarwal, S., Laradji, I. H., Charlin, L., et al. (2024). LitLLM: A toolkit for scientific literature review. *arXiv:2402.01788*.

Asai, A., He, J., Shao, R., et al. (2024). OpenScholar: Synthesizing scientific literature with retrieval-augmented LMs. *arXiv:2411.14199*.

Baek, J., Jauhar, S. K., Cucerzan, S., et al. (2025). ResearchAgent: Iterative research idea generation over scientific literature with large language models. In *Proc. NAACL*.

Beltagy, I., Lo, K., and Cohan, A. (2019). SciBERT: A pretrained language model for scientific text. In *Proc. EMNLP*.

Bouma, G. (2009). Normalized (pointwise) mutual information in collocation extraction. In *Proc. GSCL*, pages 31–40.

Grootendorst, M. (2022). BERTopic: Neural topic modeling with a class-based TF-IDF procedure. *arXiv:2203.05794*.

Hong, S., Zhuge, M., Chen, J., et al. (2024). MetaGPT: Meta programming for a multi-agent collaborative framework. In *Proc. ICLR*.

Lála, J., O'Donoghue, O., Shtedritski, A., et al. (2023). PaperQA: Retrieval-augmented generative agent for scientific research. *arXiv:2312.07559*.

LangChain, Inc. (2024). LangGraph: A library for building stateful, multi-actor applications with LLMs. `https://github.com/langchain-ai/langgraph`. Accessed: 2026-06-22.

Li, G., Hammoud, H. A. A. K., Itani, H., et al. (2023). CAMEL: Communicative agents for "mind" exploration of large language model society. In *Proc. NeurIPS*.

Liang, W., Zhang, Y., Cao, H., et al. (2024). Can large language models provide useful feedback on research papers? a large-scale empirical analysis. *NEJM AI*, 1(8).

Liu, R. and Shah, N. B. (2023). ReviewerGPT? an exploratory study on using large language models for paper reviewing. *arXiv:2306.00622*.

Lu, C., Lu, C., Lange, R. T., et al. (2024). The AI scientist: Towards fully automated open-ended scientific discovery. *arXiv:2408.06292*.

Madaan, A., Tandon, N., Gupta, P., et al. (2023). Self-refine: Iterative refinement with self-feedback. In *Proc. NeurIPS*.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proc. EMNLP*.

Salem, N. M., White, E., Bada, M., et al. (2025). GAPMAP: Mapping scientific knowledge gaps in biomedical literature using large language models. *arXiv:2510.25055*.

Shinn, N., Cassano, F., Berman, E., et al. (2023). Reflexion: Language agents with verbal reinforcement learning. In *Proc. NeurIPS*.

Si, C., Yang, D., and Hashimoto, T. (2024). Can LLMs generate novel research ideas? a large-scale human study with 100+ NLP researchers. *arXiv:2409.04109*.

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

Taylor, R., Kardas, M., Cucurull, G., et al. (2022). Galactica: A large language model for science. *arXiv:2211.09085*.

van Eck, N. J. and Waltman, L. (2010). Software survey: VOSviewer, a computer program for bibliometric mapping. *Scientometrics*, 84(2):523–538.

Wadden, D., Lin, S., Lo, K., et al. (2020). Fact or fiction: Verifying scientific claims. In *Proc. EMNLP*.

Wang, Y., Guo, Q., Yao, W., et al. (2024). AutoSurvey: Large language models can automatically write surveys. In *Proc. NeurIPS*.

Wei, J., Wang, X., Schuurmans, D., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. In *Proc. NeurIPS*.

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. John Wiley & Sons, 2nd edition.

Wu, Q., Bansal, G., Zhang, J., et al. (2023). AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv:2308.08155*.

Yao, S., Zhao, J., Yu, D., et al. (2023). ReAct: Synergizing reasoning and acting in language models. In *Proc. ICLR*.