From Extraction to Governed Memory: Multi-Agent Knowledge Graph Construction with Domain-Expert Review

Anonymous ACL submission

Abstract

Knowledge graphs used by agentic systems are often treated as flat stores of extracted triples, with little record of who owns a fact, why it was admitted, or how it should be used downstream. We argue that reliable agentic knowledge systems require governance as an essential component of graph construction to bridge this gap. We propose MAGG, a principled multi-agent framework for constructing Governed Knowledge Graphs that introduces explicit governance decisions for reliable and trustworthy knowledge sharing. A domain classifier first induces entity and relation types directly from document content, enabling operation in open-world settings without fixed schemas. Candidate triples are assigned to domain owners, reviewed against supporting evidence, admitted through governance decisions, and stored with audit metadata. The same ownership structure is reused during question answering, where queries are routed to domain-specific graph experts rather than answered through undifferentiated retrieval. Our evaluation demonstrates MAGG's effectiveness: On SciERC, MAGG improves strict triple F1 by 47% and mapped triple F1 by 51% over flat insertion. A blinded review of 120 triples finds governed-only triples more often source-supported than flat-only ones, and revised triples supported in 100% of cases. Finally, on MuSiQue, MAGG outperforms Microsoft GraphRAG by 9.0 exact-match points and 11.2 token-F1 points.

1 Introduction

Knowledge graphs are increasingly used as memory substrates for agentic systems, but most current pipelines still treat them as flat stores of extracted triples. While other approaches have explored multi-agent coordination for knowledge graph construction (Lu et al., 2025; Ye et al., 2024), our system proposes a novel framework that integrates governance into the knowledge graph construction process from inception. This leaves several practical questions unanswered: 1) who owns a fact? 2) what evidence justified its inclusion? 3) how should conflicting updates be resolved and 4) how should downstream applications decide which part of the graph to consult? A flat graph can store relations, but it does not by itself provide the organizational structure needed for controlled updates, auditability, or domain-aware reasoning.

These limitations become more visible when knowledge graphs are used by active systems rather than passive analytics pipelines. Agentic systems must continually ingest new information, decide whether candidate facts should be admitted, revise facts when better evidence appears, and route queries to the right portion of the graph. In this setting, a graph is not merely a repository of extracted relations, but a memory structure that must support ownership, evidence-grounded review, incremental update, and downstream use. We argue that governance is therefore not an optional add-on, but a necessary graph layer for reliable agentic knowledge systems.

To this end, we developed MAGG as a knowledge representation system in which facts are admitted through explicit ownership, review, and provenance decisions. This organizational layer enables the same graph structure to support controlled construction, incremental updates, audit, and domain-routed reasoning. Furthermore, MAGG's governance is exercised by LLM-based *domain experts*: specialized agents that own domain-specific subgraphs, review candidate facts against evidence along with local graph context, and control admission into memory. We refer to these agents simply as Domain Expert Agents, which are LLM-based reviewers, not human experts.

MAGG is organized into three coordinated layers. The *creation layer* processes documents through a multi-stage extraction pipeline with blackboard-based deliberation for uncertain enti-

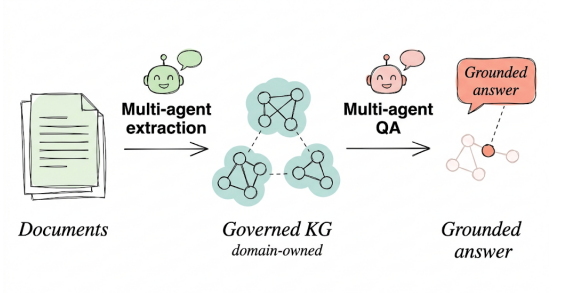

Figure 1: Our framework parses documents into a governed knowledge graph through multi-agent deliberation and answers user queries through domain-expert collaboration over the resulting graph.

ties, relations, and triples. The *governance layer* routes candidate triples to Domain Expert Agents, which review them against evidence and local domain memory and record each admission decision. The *application layer* builds downstream functionality on top of the governed graph, including domain-routed question answering and incremental self-organization as new documents are processed. Figure 1 summarizes this flow, from raw documents through governed construction to domain-expert question answering. **Our central architectural idea is that governs what enters memory also determines how memory is later accessed and updated.**

We evaluate MAGG in both fixed-schema and open-world settings. These results support the broader claim of the paper: knowledge graphs for agentic systems should not be treated as flat collections of extracted triples, but as governed memory structures in which every fact has an owner, evidence, an admission path, and an audit trail. Experimental results are presented in §4.

The contributions of this work are threefold. First, we formalize Governed Knowledge Graphs as a representation that augments entities and triples with domain ownership, explicit governance decisions, and LLM agent-mediated review. Second, we introduce MAGG, a multi-agent construction pipeline to operationalize these needs with governed admission and domain-routed Q/A. Third, we evaluate governance through extraction quality, governance ablations, human triple review, and downstream Q/A and show that the resulting domain-owned graph structure is useful for downstream graph-based question answering.

2 Related Work

2.1 Knowledge Graphs & Entity Extraction

Knowledge graph construction has historically relied on supervised pipelines that divided the extraction problem into sequential tasks: named entity recognition (NER), relation classification, and coreference resolution. Early efforts to unify these stages, such as the SciERC multi-task framework (Luan et al., 2018), learned joint span representations across these extraction tasks over scientific text. With the shift toward generative architectures, these systems collapsed into single-pass operations. Cabot and Navigli (2021) framed relation extraction as a seq2seq generation problem covering over 200 relation types. Subsequent approaches, notably UIE (Lu et al., 2022) and InstructUIE (Wang et al., 2023), integrated all core information extraction tasks under a single text-to-structure paradigm. Xue et al. (2024) modularized document-level extraction into distinct generative phases for relation identification and endpoint binding, while GLiNER (Zaratiana et al., 2024) enabled zero-shot parallel span scoring for arbitrary entity types without task-specific fine-tuning. These single-model systems share a common failure mode: when faced with conflicting extractions, hallucinated entities, or low-confidence boundary conditions, they have no systematic mechanism to deliberate or resolve internal disagreement.

2.2 Multi-Agent Systems

To address the coordination and verification deficits of single models, multi-agent frameworks have introduced agentic deliberation. Ye et al. (2024) demonstrated that a three-agent collaborative topology significantly reduces extraction hallucinations compared to isolated models. Scaling this approach, KARMA (Lu et al., 2025) deployed a nine-agent architecture with specialized roles for schema alignment and conflict resolution to enrich biomedical knowledge graphs. Concurrent research into multi-agent deliberation topologies, including unstructured debate mechanisms (Du et al., 2024), structured communication protocols (Chan et al., 2024), and iterative self-refinement (Madaan et al., 2023) confirms that peer disagreement provides a more reliable calibration signal than a single model's self-reported confidence scores. Despite these architectural advances, the memory structures produced by these systems remain fundamentally flat. They rely on a central controller to merge

verified extractions into a global store, stripping away metadata regarding domain ownership, extraction provenance, and the governance rules that authorized a triple's admission.

2.3 QA Evaluation

In the context of graph-grounded retrieval, GraphRAG (Edge et al., 2024) constructs entity graphs and executes queries by map-reducing precomputed Leiden community summaries, imposing a purely topological hierarchy rather than an ownership-based one. Similarly, while iterative graph-traversal techniques (Sun et al., 2024; Luo et al., 2024) outperform flat semantic retrieval for multi-hop reasoning tasks, they treat the underlying knowledge graph as a static artifact devoid of construction provenance. Evaluation frameworks have also shifted toward granular verification. Min et al. (2023) popularized the decomposition of generated text into atomic facts validated against retrieval sources, a paradigm extended by SAFE (Wei et al., 2024) to achieve human-annotation quality via web search. RefChecker (Hu et al., 2024) sharpened the verification unit further to strict *(subject, predicate, object)* claim-triplets. LLM-as-a-judge approaches (Zheng et al., 2023) scale evaluation cheaply but exhibit known positional and verbosity biases that degrade reliability on structural claims.

None of these systems operate over a memory substrate with construction provenance. In a governed graph, the verdict on any claim should trace back to the domain owner and the governance decision that authorized its admission.

3 Methodology

3.1 Governed Knowledge Graph

Let R be the relation vocabulary, $C \subseteq E \times R \times E$ the set of candidate triples, and $T \subseteq C$ the admitted triples. A Governed Knowledge Graph is $G = (E, C, T, D, \phi, \gamma, \pi, A)$, where $\phi : E \to 2^D$ assigns entity ownership, $\gamma : C \to \Omega$ maps candidates to governance decisions, $\Omega = \{$`approve`, `reject`, `revise`, `escalate`, `auto_approve`$\}$, π stores evidence/provenance, and A is the audit log. A candidate c enters T only if $\gamma(c) \in \{$`approve`, `revise`, `auto_approve`$\}$; escalations require later approval.

This definition differs from a standard knowledge graph in two ways. First, the graph contains an explicit organizational layer through domains and ownership. Second, admission into the triple set is mediated through governance rather than direct insertion. In implementation, this is realized through a two-phase protocol: propose a triple, obtain a governance decision, and then commit that decision. This changes the graph from a passive fact store into a managed memory structure.

The ownership function is written as $\phi : E \to 2^D$, rather than $\phi : E \to D$, because one entity may legitimately belong to more than one domain. In a scientific corpus, the same entity can participate in multiple conceptual regions of the graph: a method may belong to a methods domain and also sit inside a task-specific domain; a biomarker may belong to both an inflammation domain and a cardiometabolic domain. Using 2^D means $\phi(e)$ returns a set of owners, not a single owner. This is what makes cross-domain governance possible. Once entities have domain assignments, a triple can be classified as single-owner, cross-domain, or unowned depending on the ownership of its endpoints.

Operationally, the data structure consists of more than entities and triples. A governed triple also carries an ownership assignment, a governance action, a rationale, and an audit record. In the implementation, every admitted triple passes through `propose_triple(...)` , produces a `GovernanceDecision`, and is then committed through `commit_decision(...)`. This is the concrete mechanism that instantiates γ. The graph therefore stores not only what is known, but also how that knowledge entered memory and who was responsible for it.

3.2 System Overview

The system receives a corpus of documents. A top-level orchestrator manages the creation layer, which turns documents into entities and triples; the governance layer, which controls admission into memory; and the application layer, which uses the governed graph for downstream tasks. The pipeline is mostly sequential, and deliberation is activated only when an agent signals uncertainty or conflict.

The workflow is as follows; full construction-agent prompts and tool access are provided in Appendix C.

1. `DocumentProcessor`: segments each document into coherent chunks.

2. `DomainClassifier`: derives a working schema for the document. In benchmark mode it

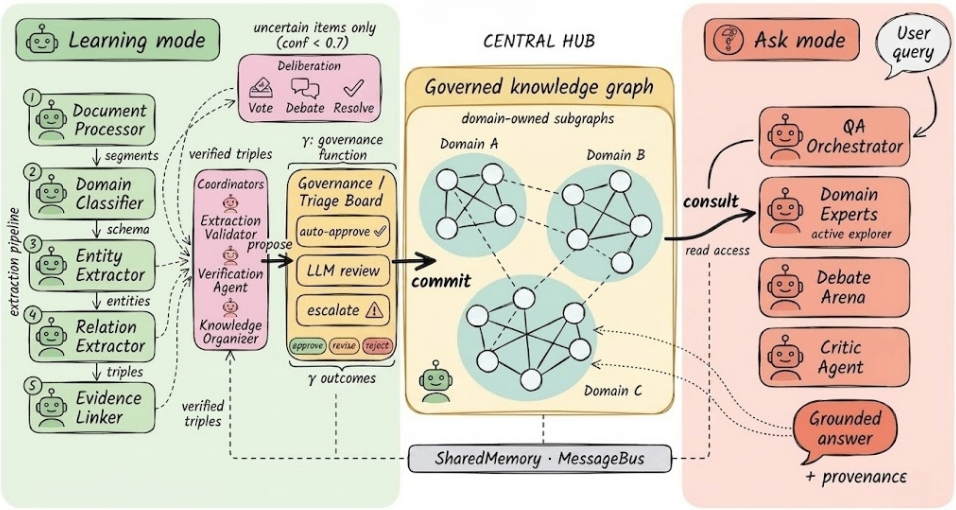

Figure 2: System architecture. The framework operates in two modes over a shared governed knowledge graph. In **Learning mode** (left), worker agents run a sequential extraction pipeline, with uncertain items routed through a deliberation side-loop before governance review and commit. In **Ask mode** (right), a QA orchestrator routes user queries to domain experts that consult the graph through active exploration before returning a grounded answer with provenance.

uses a fixed schema; in open-world mode it discovers entity and relation types from the content. From this schema, the system bootstraps preliminary domains before any triples are admitted. This is the first point at which governance becomes active during creation.

3. `EntityExtractor`: identifies entity mentions, canonical labels, candidate types, and domain assignments. The shortest-meaningful-noun-phrase rule is used only to normalize the entity span itself, so that entities are anchored to compact canonical labels rather than overly long surface forms. Relation extraction is not limited to noun pairs: downstream agents still use the full sentence context, including predicates, modifiers, and supporting evidence, to determine whether a typed relation should be proposed between extracted entities. The extractor also preserves short scientific terms, abbreviations, tool names, and task phrases rather than discarding them as generic text. These assignments partially instantiate the ownership function (ϕ), allowing the system to know who owns a proposed fact during creation.

4. `RelationExtractor`: identifies relations between current entities and produces candidate triples. In fixed-schema benchmark mode, the extractor is constrained to the benchmark label set; in open-world mode, it can propose new relation structure.

5. `EvidenceLinker`: grounds each candidate triple in the source document by retrieving explicit supporting text spans and source references. It is instructed to quote or localize evidence directly from the document rather than generate free-form justifications, and these grounded spans are then inspected by the verification and governance stages before the triple can be admitted.

6. `VerificationAgent`: removes unsupported or low-quality extractions and outputs a verified set of triples suitable for admission review.

7. `KnowledgeOrganizer`: and governance layer take over. Each verified triple is proposed to the governed graph rather than inserted directly. The system routes the triple to the relevant domain or domains, produces a governance decision, commits the decision, and appends that decision to the audit log.

Table 1 summarizes the specialized agents used during construction. The pipeline is sequential by default: each agent transforms the current representation and passes it forward, while uncertain outputs are routed through deliberation before they can proceed to governance review.

Documents are processed incrementally: later documents can resolve entities against the existing graph, while new triples are still proposed from

Agent	Input	Output	Purpose
DocumentProcessor	Documents	Chunks	Document segmentation.
DomainClassifier	Chunks	Working schema and domains	Schema/domain bootstrapping
EntityExtractor	Chunks	Entities	Mention & canonical entity detection
RelationExtractor	Entities and context	Candidate triples	Typed relation proposal
EvidenceLinker	Triples and source text	Evidence spans	Provenance grounding
VerificationAgent	Triples and evidence	Verified candidates	Quality filter
KnowledgeOrganizer	Verified candidates	Governance decisions	Admission & audit

Table 1: Specialized agents in the MAGG construction pipeline.

source evidence before being reviewed against prior domain memory. Thus, existing graph structure provides context for review but does not deterministically force new facts into old relation patterns.

3.3 Deliberation, Blackboard, and Voting

The pipeline is sequential by default; deliberation is an exceptional path triggered when an agent's confidence falls below a threshold of 0.7, chosen to balance precision and compute efficiency, and serves as a middle ground between excessive automatic review and deliberation.

Rather than discarding all low-confidence items, the system holds them as *hypotheses* for peer review, because many valuable facts are locally ambiguous rather than incorrect. Abbreviations, method names, and relation labels often require neighboring evidence to resolve. Deliberation lets the system repair uncertain candidates before governance review, rather than either admitting unstable triples or losing useful facts.

A hypothesis is a structured claim about an extraction unit that is not trusted enough for automatic continuation, typically an uncertain entity, triple, or relation type, including its proposed content, posting agent, initial confidence, and supporting evidence. For example, from the sentence "We present Minimum Bayes-Risk (MBR) decoding for statistical machine translation," the system may propose the candidate triple (`Minimum Bayes-Risk`, `Used-for`, `statistical machine translation`) at confidence 0.62. The subject span is under-specified relative to the source text, which refers more precisely to `Minimum Bayes-Risk decoding`, so the candidate is held in hypothesis state rather than admitted or discarded. This intermediate state is necessary because many low-confidence cases are not invalid, rather they are only locally ambiguous, and resolving them immediately preserves document context without polluting the graph with unstable structure.

The blackboard is a short-lived shared-memory workspace storing transient extraction state: pending hypotheses, vote requests, and vote rationales. We describe the shared memory architecture in more detail in Appendix B. A DELIBERATIONCOORDINATOR polls three role-conditioned voters–ENTITYEXTRACTOR (entity plausibility), RELATIONEXTRACTOR (relation coherence), and EVIDENCELINKER (textual support)–and aggregates a keep/no-keep decision. These differ from the Domain Expert Agents who govern admission; here, votes reflect extraction reliability rather than topical ownership. If votes conflict, the coordinator triggers a short structured debate. The final decision and rationale are written back to shared memory and used in two ways: operationally, as domain-local memory for later reviewers, and analytically, as evidence for debugging, ablation, and human validation. Accepted or revised candidates then proceed to governance review, where the governed graph stores only persistent, evidence-reviewed knowledge.

3.4 Governed Graph Use

MAGG relies on the governed knowledge graph as its foundation, providing a structured platform for downstream applications to build upon. We designed MAGG to support a range of applications and here we highlight two: question-answering, and incremental enrichment.

Downstream applications operate over the governed graph rather than over raw extracted text alone. For QA, an orchestrator reads the domain layout, routes questions to relevant domain experts, retrieves from their owned subgraphs and evidence, and synthesizes a final answer. For incremental enrichment, new documents are processed through the same ownership, evidence, and admission protocol, with graph updates under governed review.

4 Results

We organize the results around four questions: whether governance improves admitted graph quality, whether that gain is explained by simple filtering, whether human review confirms that governed triples are better supported, and whether the governed graph improves downstream Q/A.

4.1 Experimental Setup

To answer the preceding questions, we evaluate MAGG across three empirical settings. The first setting combines extraction quality and governance ablation because both ask how candidate triples should enter memory.

1. **Extraction Quality and Governance Ablation:** We benchmark extraction performance against the SciERC test split and isolate the impact of governed admission over direct insertion, source-grounded insertion, and single-reviewer baselines. MAGG operates in a fixed-schema mode to compute exact precision/recall metrics (Entity F1, Strict/Mapped Triple F1) against a standardized target ontology.

2. **Human Triple Review:** Because SciERC is a closed-world annotation, we conduct a blinded human adjudication study to evaluate actual source-support and graph utility for out-of-ontology triple proposals.

3. **Downstream QA:** We test end-to-end multi-hop retrieval in an open-world setting using a 100-question slice of the MuSiQue dataset (Trivedi et al., 2022), directly comparing our domain-routed QA approach against Microsoft GraphRAG.

4.2 Extraction Quality and Governance Ablation

This subsection answers the first two results questions: whether governed admission improves the quality of the stored graph, and if the improvement is explained by simpler filtering. We evaluate all conditions on the full 100-document SciERC test split using GPT-5 and the fixed SciERC schema. The documents, target ontology, and base model are held fixed; only the admission pathway changes.

We compare four admission pathways. **Direct flat insertion** inserts extracted triples after basic structural cleanup, with no evidence linking, verification, domain ownership, or review. **Source-grounded flat insertion** adds EvidenceLinker and VerificationAgent, so triples are attached to source spans and checked for support before flat insertion. **Global reviewer** adds a single centralized LLM approve/reject reviewer, but no domain ownership or local graph memory. **MAGG** routes candidate triples to Domain Expert Agents, which review evidence and local graph context before approving, rejecting, or revising the triple for admission.

We report strict and mapped triple precision, recall, and F1 in Table 2. Strict scoring requires exact subject–relation–object agreement; mapped scoring applies relation canonicalization and fuzzy entity normalization before scoring. Precision captures how clean the admitted graph is, while recall captures how much of the benchmark graph is recovered. This distinction matters because direct insertion can improve recall by admitting many candidates, but at the cost of a noisier graph.

Source grounding alone does not explain the gain: source-grounded flat insertion barely moves strict F1 (0.106 $\rightarrow$ 0.107) while admitting nearly the same number of triples. A centralized global reviewer is stronger, reducing the graph to 1599 triples and improving strict F1 to 0.143, confirming that review helps but generic review is not the full mechanism.

MAGG performs best overall. It admits 1,077 triples, achieves the highest strict F1 (0.156) and mapped F1 (0.290), and has the strongest precision under both scoring regimes. Compared with direct flat insertion, strict precision rises from 0.066 to 0.129 and mapped precision rises from 0.119 to 0.240. The net effect is a cleaner admitted graph, with strict triple F1 improving by 47% and mapped triple F1 by 51% over direct GPT-5 insertion.

This supports the governance claim. The improvement is not reproduced by source grounding alone and is only partially reproduced by a single global reviewer. MAGG adds ownership, routing, domain-local memory, and subgraph-aware review, so Domain Expert Agents evaluate triples relative to the local graph context and prior accepted or rejected relation patterns. Because SciERC is a closed benchmark rather than an exhaustive fact inventory, some source-supported triples may still be counted as false positives; the human review study below complements this benchmark by checking whether such triples are actually supported.

Condition	Entities	Triples	Strict P	Strict R	Strict F1	Mapped P	Mapped R	Mapped F1
Direct flat insertion	1742	2881	0.066	0.271	0.106	0.119	0.490	0.192
Source-grounded flat insertion	1838	2912	0.067	0.277	0.107	0.123	0.510	0.198
Global reviewer	1755	1599	0.103	0.234	0.143	0.194	0.444	0.271
MAGG	1731	1077	0.129	0.199	0.156	0.240	0.369	0.290

Table 2: Extraction quality and governance ablation on the full 100-document SciERC test split. All rows use GPT-5 and the fixed SciERC schema. Mapped scores apply relation-label canonicalization and fuzzy entity normalization before scoring.

4.3 Human Triple Review

To assess whether governance changes the quality of admitted triples rather than merely changing graph structure, we conducted a lightweight human validation study over a blinded sample of 120 triples. The sample was randomly shuffled and drawn from four mutually exclusive categories: 40 triples admitted only by the governed graph, 40 triples admitted only by the FlatKG baseline, 20 triples admitted by both systems, and 20 triples revised by the governance layer before admission. During annotation, labels were hidden so that annotators did not know which of the four categories it came from.

Annotation instructions. Annotators were instructed to judge each triple using only the provided source evidence and not to rely on outside knowledge. For each triple, annotators evaluated four properties: whether the source evidence supported the triple, whether the relation label was correct, whether the subject and object endpoints were correct, and whether the triple was useful for graph-based question answering or retrieval. Source support was labeled as *yes*, *partial*, *no*, or *unclear*. Relation correctness and endpoint correctness were labeled as *yes*, *partial*, or *no*. Usefulness was labeled as *yes*, *maybe*, or *no*. A triple was considered useful if it encoded a relation likely to help graph search, multi-hop reasoning, domain routing, or source-grounded QA. We provide the exact prompt in Appendix E.

Adjudication process. Two annotators judged whether each candidate triple was supported by the source evidence, whether its relation and endpoints were correct, and whether it was useful for graph-based QA. Disagreements were resolved by a third adjudicator. When labels differed, the adjudicated label reflected the most defensible interpretation of the evidence: direct textual support was required for a *yes* label, plausible but incomplete or slightly under-specified support was marked *partial*, and claims not grounded in the evidence were marked *no*. We report both inter-annotator agreement (Table 3) and the final adjudicated labels.

Annotation Field	Agreement
Source support	85%
Relation correctness	75%
Endpoint correctness	83%
KG/QA usefulness	88%

Table 3: Inter-annotator agreement for the blinded human validation study. Agreement was computed before adjudication over the 120 sampled triples.

The results support the governance hypothesis. Governed-only triples were judged supported or partially supported in 85.0% of cases, compared with 62.5% for FlatKG-only triples. They were also more often useful or potentially useful for QA, 77.5% versus 62.5%. Shared triples were similarly strong, with 85.0% support and 85.0% usefulness, suggesting that triples admitted by both systems form a high-confidence core. Most notably, revised triples were judged supported or partially supported in 100.0% of cases and useful or potentially useful in 95.0% of cases. Our results indicate that the governance layer does more than reject low-quality triples and emphasizes that the revision pathway is not merely rejecting bad triples; it repairs candidate facts into more supportable graph entries. We summarize the results of this experiment in Table 4.

These findings complement the automatic SciERC evaluation. While benchmark triple matching penalizes any triple absent from the gold annotation, human validation shows that many governed-only triples are source-supported and useful for downstream graph reasoning. Governance therefore appears to improve the admitted graph not only by reducing unsupported triples, but also by shaping the graph toward more usable relational facts.

4.4 Downstream QA on MuSiQue

To evaluate whether the governed graph is useful for downstream applications, we test it on

Triple Group	n	Supported or Partial	Unsupported	Useful or Maybe Useful
Governed Only	40	**85.0% (34)**	12.5% (5)	**77.5% (31)**
FlatKG Only	40	62.5% (25)	37.5% (15)	62.5% (25)
Both	20	85.0% (17)	15.0% (3)	85.0% (17)
Revised	20	**100.0% (20)**	**0.0% (0)**	**95.0% (19)**

Table 4: Adjudicated human validation results over a blinded sample of 120 triples. Governed-only triples were more often source-supported and useful for graph-based QA than FlatKG-only triples. Revised triples were judged supported or partially supported in all sampled cases, suggesting that governance can repair candidate triples before admission rather than merely filtering them. Note: percentages may not sum to 100 where adjudicated labels were marked unclear.

MuSiQue, a multi-hop question answering benchmark (Trivedi et al., 2022) designed so that answering a question requires combining facts across multiple pieces of evidence rather than retrieving a single local span. In contrast to the SciERC experiments, this setting is open-world: MuSiQue does not impose a fixed benchmark schema, so MAGG must organize entities, relations, and domains directly from unconstrained text.

We use a 100-question slice from MuSiQue 2-hop and report the results in Table 5. In this setting, each question requires linking two supporting facts. For our experiment, the supporting-only corpus contains 200 contexts drawn from the question support sets. From this corpus, MAGG first constructs a governed graph in open-world mode by extracting entities and relations from the passages, organizing them into domain-owned subgraphs, and admitting facts through its governance pipeline. We then build a QA system on top of this graph: each question is decomposed and routed to the most relevant domain experts, those experts answer from their local subgraphs and neighboring evidence, and a final synthesis step combines their responses into a single answer with source verification. We compare this domain-routed QA system against official Microsoft GraphRAG and a flat KG-QA baseline. The Flat KG-QA condition uses the same QA orchestrator, query decomposition, and synthesis stack as MAGG, but operates over a flat-admitted graph with no domain ownership or governance review. The full QA agent roles and prompts are detailed in Appendix D.

System	EM	F1	Interpretation
Flat KG-QA	0.38	0.48	Flat graph, same QA orchestrator
GraphRAG	0.42	0.53	Official graph-aware baseline
MAGG domain QA	**0.51**	**0.65**	Governed graph with domain-routed QA

Table 5: MuSiQue QA, 100-Question Slice (GPT-5).

The 0.380 to 0.510 EM gain from Flat KG-QA to MAGG isolates the governance contribution: the QA stack is identical, while the underlying graph changes from flat-admitted to governed. The 0.420 to 0.510 EM gain over GraphRAG further shows that MAGG's governed graph plus domain-routed QA outperforms a graph-aware baseline with its own retrieval approach. The same pattern holds for token F1, where MAGG reaches 0.646 compared with 0.480 for Flat KG-QA and 0.534 for GraphRAG. These results suggest that the ownership structure of the graph is useful at answer time: questions are routed through domain-owned subgraphs, domain experts return subgraph-grounded answers, and the final response is synthesized from these local views.

5 Conclusion

We introduced MAGG, a framework for constructing Governed Knowledge Graphs in which candidate triples are admitted through ownership, evidence review, governance decisions, and audit records rather than flat insertion. The same ownership structure is reused downstream for domain-routed question answering and incremental graph update.

Across SciERC, governance improves admitted triple quality over flat insertion, and ablations show that the gain is not explained by source filtering or a generic global reviewer alone. A blinded human review further suggests that governed and revised triples are more often source-supported and useful. On MuSiQue, MAGG improves over a graph-based QA baseline in a supporting-only multi-hop setting. These results support the central claim that governance is not merely a post-hoc filter, but a graph-level mechanism for organizing knowledge creation, update, audit, and downstream reasoning.

Limitations

MAGG improves admitted graph quality, but it does not solve extraction. Strict triple F1 remains modest on SciERC, reflecting the difficulty of exact triple matching and remaining errors in normalization, relation labeling, and evidence grounding. The system also introduces additional computational cost because domain-routed governance and domain-local memory require more LLM calls and prompt tokens than flat insertion. All experiments use GPT-5; whether governance gains transfer to other LLM families remains open. Finally, the open-world and human validation results are still preliminary: MuSiQue demonstrates open-world use, but we do not separately evaluate induced domain quality, and the blinded triple review should be expanded across more annotators, corpora, and domains.

References

Pere-Lluís Huguet Cabot and Roberto Navigli. 2021. REBEL: Relation extraction by end-to-end language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2370–2381. Association for Computational Linguistics.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2024. ChatEval: Towards better LLM-based evaluators through multi-agent debate. In *The Twelfth International Conference on Learning Representations*.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 11850–11875. PMLR.

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

Xiangkun Hu, Dongyu Ru, Lin Qiu, Qipeng Lu, Tianhao Zhao, Yuchuan Zhang, Yue Guo, Nan Duan, Zheng Zhang, and Alan Yuille. 2024. RefChecker: Reference-based fine-grained hallucination checker and benchmark for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 6934–6958. Association for Computational Linguistics.

Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. 2022. Unified structure generation for universal information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5755–5772. Association for Computational Linguistics.

Yuxing Lu, Wei Wu, Xukai Zhao, Rui Peng, and Jinzhuo Wang. 2025. KARMA: Leveraging multi-agent LLMs for automated knowledge graph enrichment. *arXiv preprint arXiv:2502.06472*.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232. Association for Computational Linguistics.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100. Association for Computational Linguistics.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Shuzheng Wang, Chen Lin, Yeyun Gong, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Xiao Wang, Weikang Zhou, Can Zu, Han Xia, Tianze Chen, Yuansen Zhang, Rui Zheng, Junjie Ye, Qi Zhang, Tao Gui, and 1 others. 2023. InstructUIE: Multi-task instruction tuning for unified information extraction. *arXiv preprint arXiv:2304.08085*.

Jerry Wei, Chengrun Yang, Xinying Song, Yifeng Lu, Nathan Hu, Dustin Tran, Daiyi Peng, Ruibo Liu, Da Huang, Cosmo Du, and 1 others. 2024. Long-form factuality in large language models. In *Advances in Neural Information Processing Systems*, volume 37.

Lilong Xue, Dan Zhang, Yuxiao Dong, and Jie Tang. 2024. AutoRE: Document-level relation extraction with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 213–225. Association for Computational Linguistics.

Hongbin Ye, Honghao Gui, Aijia Zhang, Tong Liu, Wei Hua, and Weiqiang Jia. 2024. Beyond isolation: Multi-agent synergy for improving knowledge graph construction. In *Proceedings of the China Conference on Knowledge Graph and Semantic Computing (CCKS)*.

Urchade Zaratiana, Nadi Tomeh, Pierre Holat, and Thierry Charnois. 2024. GLiNER: Generalist model for named entity recognition using bidirectional transformer. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5364–5376. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623.

A Token and Runtime Cost

We also report computational cost because MAGG trades additional governance calls for a cleaner and more auditable graph. Table 6 reports exact API call and token counts from JSONL usage logs.

The current usage logs do not separate every internal extraction sub-agent into distinct cost buckets. Entity extraction, relation extraction, evidence linking, and any deliberation triggered inside extraction are included in the extraction-build usage log. Governance review is separately logged when replayed as a domain-memory review stage.

B Shared Memory

MAGG uses shared memory to coordinate extraction within a document and to maintain consistency across documents. These functions are separated because the system must handle both short-lived local uncertainty and longer-lived accumulated knowledge. During extraction, agents need access to transient state such as pending hypotheses, intermediate votes, and active evidence spans. Across documents, the system needs persistent state such as admitted entities, aliases, domain ownership patterns, and prior governance decisions.

In practice, the memory system has four regions. **Working memory** stores the current pipeline state for the active document, including candidate entities, candidate triples, pending hypotheses, and vote requests. **Semantic memory** stores accepted graph structure and canonicalized entities. Once a candidate is revised and admitted, the canonical entity, its aliases, and the admitted triple are written to semantic memory for reuse in later extraction, governance, and QA. **Procedural memory** stores reusable review patterns derived from prior decisions, such as relation signatures that were previously accepted, rejected, or revised. **Episodic memory** stores per-document processing traces, such as what was proposed, revised, or rejected in a specific document.

These memory types are used at different timescales. Working memory is short-lived and is reused continuously during a single document pass. Semantic and procedural memory persist across documents and are consulted repeatedly during governance and downstream QA. Episodic memory is consulted less frequently, mainly when the system or a human needs to inspect how a prior decision was made. This separation is important because MAGG is not only extracting facts; it is building a governed memory that must distinguish temporary uncertainty from persistent knowledge.

Stage	Unit	LLM Calls	Prompt Tokens	Completion Tokens	Total Tokens	Calls / Unit	Tokens / Unit
SciERC flat extraction build	100 docs	735	2,503,556	436,940	2,940,496	7.35 / doc	29,405 / doc
SciERC source-grounded flat build	100 docs	801	2,490,259	655,658	3,145,917	8.01 / doc	31,459 / doc
SciERC MAGG build	100 docs	1,845	6,707,359	1,313,298	8,020,657	18.45 / doc	80,207 / doc
Domain-memory governance replay	2912 candidate triples	218	3,429,605	382,096	3,811,701	0.07 / candidate	1,309 / candidate
Global reviewer replay	2912 candidate triples	145	238,524	119,151	357,675	0.05 / candidate	123 / candidate
MuSiQue KG build	200 supporting contexts	630	6,599,190	829,591	7,428,781	3.15 / context	37,144 / context
MuSiQue final MAGG QA	100 questions	105	255,689	145,612	401,301	1.05 / question	4,013 / question

Table 6: LLM Call and Token Cost

C Construction Agent Roles and Prompts

Each construction agent in MAGG is documented below using a uniform schema (Inputs, Instruction/Prompt, Tools available, Output). Prompt text shown in italics is verbatim from the system prompt in our implementation; longer prompts are summarized. Uncertain items at any extraction stage are routed through the `DeliberationCoordinator` (Appendix C.7) before continuing; the coordinator is a side-protocol rather than one of the seven sequential pipeline stages, but it is documented in its own subsection because it is invoked by name throughout the workflow.

C.1 Pipeline Sequence

The construction pipeline executes the following order per document: (1) **DocumentProcessor**, (2) **DomainClassifier**, (3) **EntityExtractor**, (4) **RelationExtractor**, (5) **EvidenceLinker**, (6) **VerificationAgent**, and (7) **KnowledgeOrganizer**. The first agent receives raw document text; the last writes admitted entities and triples into the governed knowledge graph. The **DeliberationCoordinator** runs as a side-loop between any agent that flags sub-threshold confidence and the next pipeline stage.

C.2 DocumentProcessor

Title: *DocumentProcessor*
Role Description. Entry point of the pipeline. Ingests raw document text, segments it into semantically coherent overlapping chunks, and registers the document in shared memory for cross-document entity resolution.
Inputs.
- `context: AgentContext` carrying `document_id: str` and `text: str`
- `source_path: Optional[str]` containing the file path if available

Instruction / Prompt. No LLM call; segmentation is deterministic.
- Unicode NFC normalization and control-character stripping.
- Sentence-boundary split on `.!?`, then greedy packing into chunks of 1500–2000 characters with 150-character overlap (tail sentences re-added to the next chunk).
- Each segment enriched with positional metadata (`segment_id`, `char_start/end`, `word_count`, `is_first`, `is_last`).

Tools available.
- `SharedMemory.register_document` — records the raw document for cross-document entity resolution.
- `store_in_memory(EPISODIC, ...)` — logs the processing event.
- No LLM, no MessageBus, no deliberation.

Output. `ExtractionResult` with `items: List[Dict]` and `confidence: float` derived from segment-length quality and coverage heuristics.

C.3 DomainClassifier

Title: *DomainClassifier*
Role Description. Discovers the document's domain from scratch (no predefined taxonomy) and generates a domain-specific extraction schema—entity types, relation types, and few-shot examples mined from the text. Broadcasts the schema to downstream extraction agents.
Inputs.
- `context: AgentContext`
- `segments: List[Dict]` from `DocumentProcessor` (up to 4000 chars: first 2 + middle + last segments)

Instruction / Prompt. Two LLM calls, both asking the model to invent types rather than use predefined categories.
1. **Domain analysis.** System: *"You are an expert at discovering knowledge structures from scratch. NEVER use predefined schemas or standard taxonomies. Read the document carefully and INVENT a custom schema that fits THIS content. Types must be UPPER_SNAKE_CASE."* Returns JSON with `primary_domain`, `sub_domains`, `entity_types` `[{type, description, priority, examples_from_text}]`, `relation_types` `[{type, description, source_types, target_types, priority}]`, `few_shot_examples`, and `extraction_parameters` `{complexity, knowledge_density, requires_coreference, has_temporal_relations, ...}`. Called with `self_consistency(n_samples=3, temperature=0.4)`.
2. **Relation examples.** Given the discovered types, extract one concrete example sentence with subject/object per relation type.

Tools available.
- LLM: `call_llm`, `call_llm_with_self_consistency`.
- `SharedMemory.store_in_memory(SEMANTIC, ...)` — caches domain context.
- `MessageBus.send_message(INFORM, ...)` — broadcasts entity types to `EntityExtractor` and relation types to `RelationExtractor`.
- `escalate_to_coordinator` when confidence is below threshold.

Output. `ExtractionResult` with `items = [domain_config]`. `context.domain` is set to `primary_domain` for downstream agents. A generic fallback schema is returned if LLM parsing fails.

C.4 EntityExtractor

Title: *EntityExtractor*
Role Description. Four-stage entity extraction with integrated coreference resolution. Discovers entity types beyond the domain schema when needed, registers cross-document aliases, and submits low-confidence entities to deliberation.
Inputs.
- `context: AgentContext`
- `segments: List[Dict]` from `DocumentProcessor`
- `domain_config: Dict` from `DomainClassifier`
- Listens on `MessageBus` for `INFORM` messages carrying `entity_types`.

Instruction / Prompt. Four sequential JSON-constrained stages.
1. **Initial extraction.** *"Extract entities based on what you observe in the text, not predefined categories. Err on the side of inclusion. Exclude generic dates, bare numbers, common adjectives."* Output: `[{text, start, end, type_guess}]`.
2. **Boundary refinement** (batches of 30). *"You are an expert at identifying precise entity boundaries."* Output: `[{text, original_text, start, end, boundary_fixed}]`.
3. **Type assignment** (batches of 25). Assigns discovered types in `UPPER_SNAKE_CASE`. Returns `type`, `type_confidence`, `type_reasoning`. Combined confidence = `(stage1_conf + type_conf + self_consistency_conf) / 3`.
4. **Coreference resolution** (batches of 20). Given the current batch and up to 20 known entities from KG/memory, groups mentions into canonical entities; aliases registered via `SharedMemory.register_entity_alias`.

Self-consistency (`n_samples=3`) is applied to stages 1 and 3 when enabled.
Tools available.
- LLM: `call_llm`, `call_llm_with_self_consistency`.
- `SharedMemory.register_entity_alias`, `add_entity_context`, `store_in_memory(SEMANTIC)`.
- `KnowledgeGraph` (read): loads up to 50 known entities for coreference.
- `MessageBus.receive_messages` (domain info).
- Deliberation: `submit_for_deliberation(hypothesis_type="entity", ...)`; `evaluate_hypothesis_for_vote` provides voting logic.
- `escalate_to_coordinator`.

Output. `ExtractionResult` with `items: List[Dict]` carrying `text`, `type`, `confidence`, `aliases`, and span metadata.

C.5 RelationExtractor

Title: *RelationExtractor*
Role Description. Three-stage Relation–Head–First (RHF) relation extractor. Identifies which relation types are present in each segment, binds heads (subjects) for each relation, then resolves tails (objects). Constrained to the benchmark label set in fixed-schema mode; in open-world mode it can propose new relation structure.
Inputs.
- `context: AgentContext`
- `segments: List[Dict], entities: List[Dict], domain_config: Dict`
- Listens on `MessageBus` for `INFORM` messages carrying `relation_types`.

Instruction / Prompt. Three sequential LLM calls per segment, plus optional pairwise gleaning.
1. **Stage 1 — Relation identification.** Given the segment and entity list, return the subset of relation types from the schema that the segment plausibly expresses, each with a one-sentence rationale.
2. **Stage 2 — Head binding.** For each identified relation, return `head_bindings: [{relation, head_entity_id, head_text, evidence_span}]`. A short *direction-only* hint is appended to the prompt to discourage subject/object inversion (e.g., for `used-for`, the head is the method being used).
3. **Stage 3 — Tail binding** (batches of 30 head bindings). For each head binding, identify the OBJECT (tail) entity. *"If you cannot find a valid, distinct object entity, SKIP that head binding entirely."*

Optional pairwise relation gleaning runs after Stage 3 for additional recall when enabled.
Tools available.
- LLM: `call_llm`.
- `SharedMemory.store_in_memory(WORKING, ...)`.
- `MessageBus.receive_messages` (relation type schema).
- Deliberation: `submit_for_deliberation(hypothesis_type="relation_type" | "triple", ...)`.
- Schema enforcement: `_enforce_fixed_schema_relations` drops triples whose label is outside the fixed schema.

Output. `ExtractionResult` with `items` as candidate triples `{subject, relation, object, confidence, evidence_segment, stage_diagnostics}`. `metadata.relation_funnel` records counts at each RHF stage for downstream diagnostics.

C.6 EvidenceLinker

Title: *EvidenceLinker*
Role Description. Attaches sentence-level evidence to each triple, classifies evidence as explicit / implicit / inferred, and optionally cross-references against prior KG knowledge to adjust confidence. Submits weak-evidence triples to deliberation.
Inputs.
- `context: AgentContext`
- `triples: List[Dict]` from `RelationExtractor`
- `segments: List[Dict]` from `DocumentProcessor`

Instruction / Prompt. Two LLM stages.
1. **Evidence linking.** *"Find supporting sentence(s). Label* `evidence_type` $\in$ *{explicit, implicit, inferred}. Set* `evidence_strength` $\in [0, 1]$. *List any contradicting text with char positions."* Batch size 3–20 (adaptive to token budget).
2. **Cross-reference** (optional, requires `enable_cross_reference`). *"Compare new triples against up to 30 prior triples from KG/memory. Assign* `consistency_status` $\in$ *{supported, contradicted, novel, refined} and* `confidence_adjustment` $\in [-0.3, +0.3]$*."*

Final confidence is computed deterministically:

$$\begin{aligned}
\text{final} &= \text{base_confidence} \\
&\quad \times \text{evidence_multiplier} \\
&\quad \times \text{evidence_strength} \\
&\quad + \text{xref_adjustment} \\
\text{evidence_multiplier} &= \{\text{explicit: } 1.0, \\
&\quad \text{implicit: } 0.85, \\
&\quad \text{inferred: } 0.7\} \\
\text{xref_adjustment} &= \{\text{supported: } +0.15, \\
&\quad \text{contradicted: } -0.3, \\
&\quad \text{novel: } 0, \\
&\quad \text{refined: } +0.1\}
\end{aligned}$$

Tools available.
- LLM: `call_llm`.
- `SharedMemory.store_in_memory(SEMANTIC, evidence_linked_triples=...)`, `retrieve_from_memory`.
- `KnowledgeGraph` (read): pulls up to 50 prior triples.
- Deliberation: `submit_for_deliberation(hypothesis_type="triple")` for weak-evidence triples; `evaluate_hypothesis_for_vote` provides evidence-based votes (`strong_accept` for explicit+strong evidence, `reject` for contradictions).
- `escalate_to_coordinator`.

Output. `ExtractionResult` with `items` carrying `evidence_type`, `evidence_strength`, `supporting_text`, optional `consistency_status`, and adjusted `confidence`.

C.7 DeliberationCoordinator

Title: *DeliberationCoordinator* (protocol, not a `BaseAgent`)

Role Description. Side-loop between extraction and validation. Orchestrates multi-agent voting and structured debate over entities and triples whose confidence falls in the uncertain band $0.35 \leq c < 0.7$. Items outside the band are accepted (≥ 0.7) or rejected (< 0.35) without a vote.

Inputs.

- `entities: List[Dict]`, `triples: List[Dict]` from upstream agents
- `segments: List[Dict]`, `context: AgentContext`

Instruction / Protocol. For each uncertain item:

1. Submit a `Hypothesis(author, type∈{entity, triple, relation_type}, content, confidence, evidence)` to the blackboard.
2. Broadcast vote requests (`REQUEST`, `HIGH` priority) to the voting panel: `{EntityExtractor, RelationExtractor, EvidenceLinker}` by default.
3. Collect votes on a 7-level scale: `STRONG_ACCEPT` (+1.0), `ACCEPT` (+0.75), `WEAK_ACCEPT` (+0.5), `ABSTAIN` (0), `WEAK_REJECT` (−0.5), `REJECT` (−0.75), `STRONG_REJECT` (−1.0).
4. Weighted score = $\sum$(vote_weight × voter_confidence × agent_weight) with `AGENT_WEIGHTS` (`VerificationAgent`=1.5, `EvidenceLinker`=1.2, `EntityExtractor` = `RelationExtractor` = 1.0, `DomainClassifier`=0.8).
5. Debate is triggered if |weighted_score| < 0.3: agents contribute support / oppose arguments over multiple rounds before final resolution.
6. Resolution: `min_votes=2`, `consensus_threshold=0.6`.

Tools available.

- `SharedMemory.post_to_blackboard`, `get_blackboard_entries`, `resolve_blackboard_entry`.
- `MessageBus.send` (vote and debate requests), `receive`.
- Per-agent delegation: `evaluate_hypothesis_for_vote`, `provide_debate_argument`.

Output. Updated `entities` / `triples` lists containing items that survived voting or were high-confidence; counters `voting_sessions`, `debates_triggered`, `items_accepted_by_vote`, `items_rejected_by_vote`.

C.8 VerificationAgent

Title: *VerificationAgent*
Role Description. Quality filter that turns the evidence-linked candidate set into the verified set passed to the `KnowledgeOrganizer`. The agent first runs an iterative validate–refine loop to repair fixable issues in the candidate entities and triples, then runs a final source-grounding verification pass that drops unsupported or hallucinated triples and optionally checks cross-document consistency against the existing graph. The agent also processes blackboard escalations from upstream workers and participates in debates orchestrated by the `DeliberationCoordinator` (Appendix C.7).
Inputs.
- `context: AgentContext` (carrying source text)
- `entities: List[Dict], triples: List[Dict]` from `EvidenceLinker`, post-deliberation
- Blackboard escalations from any upstream worker
- Listens on `MessageBus` for `DELEGATE` with `action="verify"`.

Instruction / Prompt. Three LLM stages plus deliberation participation.
1. **Validation.** *"You are an expert extraction validator."* Given source text (up to 6K chars), entities, and triples, return per-item `{valid, adjusted_confidence, issues, corrections}` plus overall `quality` $\in [0, 1]$ and `recommendations`. Batch 40.
2. **Refinement.** *"Apply corrections precisely."* Emits `refined_entities, refined_triples, quality_after_refinement`. Batch 30. The validate–refine pair iterates while `overall_quality` $<$ `quality_threshold` and `iteration` $<$ `max_refinement_iterations` (default 4).
3. **Source verification.** *"Accept BOTH explicit and reasonably inferred relationships. Only reject triples clearly contradicted or completely unsupported by the text. Partial support counts as valid with lower confidence."* Per triple, return `verification_status` $\in$ {verified, partial, rejected, hallucinated}, `final_confidence`, `supporting_evidence` (exact quote), and optional `rejection_reason`. Batch 30. When prior KG triples exist, an additional cross-document consistency pass classifies each new triple as {`consistent`, `contradicts`, `refines`, `redundant`} and recommends `action` $\in$ {`add`, `update`, `reject`, `merge`}.
4. **Deliberation participation.** When invoked by the `DeliberationCoordinator`, the agent (a) maps {`valid`, `confidence`} to a `VoteType` (e.g. `valid=True, conf`$\geq$`0.8` $\rightarrow$ `STRONG_ACCEPT`; `conf`$\leq$`0.2` $\rightarrow$ `STRONG_REJECT`) and (b) when prompted for a debate turn, returns {`position, argument, key_points`} for the candidate under review.

Decision logic: `verified` $\rightarrow$ kept; `partial` $\rightarrow$ kept unless `strict_mode=True`; `rejected` or `hallucinated` $\rightarrow$ dropped; remaining triples must satisfy `final_confidence` $\geq$ `quality_threshold` (0.45).
Tools available.
- LLM: `call_llm` for validation, refinement, and verification.
- `SharedMemory.get_blackboard_entries(type="escalation"), resolve_blackboard_entry, store_in_memory(WORKING, ...)`.
- `KnowledgeGraph` (read): up to 100 prior triples for cross-document consistency.
- `MessageBus.receive_messages` (`DELEGATE` escalations); `send DELEGATE` with `action="integrate"` to `KnowledgeOrganizer`.
- Deliberation: `process_pending_deliberations, cast_vote, provide_debate_argument, get_deliberation_results()`.

- `DebugLogger.log_decision` for every accept/reject.

Output. `ExtractionResult` where each approved triple carries `verification_status`, `final_confidence`, `supporting_evidence`, optional `consistency_status` and `consistency_action`; `metadata.refinement_iterations` reports validate–refine loops run, and `metadata.verification_summary` reports {total, verified, partial, rejected, hallucinated} counts.

C.9 KnowledgeOrganizer

Title: *KnowledgeOrganizer*

Role Description. Final integration stage. Deduplicates entities, normalizes relation names, resolves triple arguments to canonical entity IDs, and writes everything into the shared `KnowledgeGraph`. This is also the agent that proposes candidate triples to the governance layer in governed mode.

Inputs.

- `context: AgentContext`
- `entities: List[Dict]` — verified entities from `VerificationAgent`
- `triples: List[Dict]` — approved triples only
- Listens on `MessageBus` for `DELEGATE` with `action="integrate"`.

Instruction / Prompt. Two LLM calls plus deterministic integration.

1. **Entity deduplication.** (a) Rule-based pass merges exact case-insensitive duplicates. (b) LLM pass: *"You are an expert at entity resolution."* Returns `merge_groups[ {canonical_id, canonical_name, merge_ids, reason}]`. Aliases registered in `SharedMemory.entity_aliases`.
2. **Relation normalization.** *"Normalize relation names consistently."* Returns `normalizations[ {original, normalized, is_inverse, reason}]`. Cache stored in `self.relation_mappings` to avoid re-querying.
3. **KG integration** (deterministic):
   - Filter garbage entities (empty, pure numeric, trivial pronouns like "we", "it", "this study", length < 2).
   - Build `name_to_id` lookup from entity text, IDs, aliases, existing KG labels, and shared-memory aliases (prevents phantom entities).
   - Resolve subject/object strings to existing entity IDs; if unresolved, create a new entity of type `UNRESOLVED` rather than dropping the triple.
   - Call `KnowledgeGraph.add_entity(...)` and `add_triple(...)`.

Tools available.

- LLM: `call_llm` for dedup and normalization.
- `KnowledgeGraph` (write): `add_entity`, `add_triple`.
- `SharedMemory.register_entity_alias`, `store_in_memory(SEMANTIC, ...)`.
- `MessageBus.receive_messages` for incoming `integrate` delegation.
- Reporting: `get_kg_stats()`, `export_knowledge_graph()`.

Output. `ExtractionResult` with `metadata.kg_stats` = {`total_entities`, `total_triples`, `entity_types`, `relation_types`, `unique_relations`} and `metadata.integration_stats` = {`entities_added`, `entities_merged`, `triples_added`, `triples_updated`, `relations_normalized`}. *Side effect:* the shared `KnowledgeGraph` is mutated and can be exported as JSON.

D QA Agent Roles and Prompts

The application layer reuses the governed graph for downstrresizeboxeam question answering. Three agents participate in each query: the **QAOrchestrator**, which decomposes a user question and routes it to one or more **DomainExpertAgents**; the **DomainExpertAgents**, which answer from their owned subgraphs; and a **SynthesisAgent**, which combines the per-domain answers into a single grounded response. Each agent is documented below using the same schema (Inputs, Instruction/Prompt, Tools available, Output) as the construction agents in Appendix C.

D.1 QAOrchestrator

Title: *QAOrchestrator*

Role Description. Top-level coordinator of question answering. Reads the domain layout of the governed graph, decomposes the user question into sub-questions, routes each sub-question to one or more domain experts, gathers cross-domain bridge context, and invokes synthesis.

Inputs.
- `question: str`
- `governed_kg` — the persistent governed graph and its domain layout

Instruction / Prompt. Single LLM call for decomposition and routing.

1. **Decomposition and routing.** System: *"You are a query decomposition and routing expert. Prefer the fewest domains needed and return only valid JSON."* User prompt: *"You are a query routing agent. Given a user question and a list of available domain experts, decompose the question into sub-questions and route each to the most relevant domain expert(s). If the question is simple and maps to a single domain, return just one sub-question."* The prompt embeds a summary (label, description, owned topics) for every domain in the org chart.
2. **Output schema** returned by the LLM:

```
{"sub_questions":
[{"question": ...,
"target_domains": [...],
"context": ...}]}
```

Tools available.
- LLM: `chat_completion_json` for decomposition and routing.
- Org-chart access for the available-domain summary used in the routing prompt.
- `DomainExpertAgent` dispatch (one call per routed domain).
- Cross-domain context assembly across the answers returned by domain experts before synthesis.
- `SynthesisAgent` (Appendix D.3) for final answer assembly.

Output. A combined record carrying the original question, the decomposed sub-questions and their routed domains, the per-domain expert responses, the synthesised `final_answer` and `final_answer_short`, and overall coverage, confidence, and any remaining knowledge gaps.

D.2 DomainExpertAgent

Title: *DomainExpertAgent*
Role Description. Answers a sub-question from a single governed subgraph. Each agent owns one domain, retrieves a query-focused slice of its subgraph, optionally finds multi-hop paths or short neighbourhoods around query entities, and produces an evidence-grounded answer with explicit triple citations.
Inputs.

- `query: str`, optional `context: str` (routing hints or bridge context from the orchestrator)
- `domain: Domain` — the owned subgraph
- `full_kg: KnowledgeGraph` — read-only access for multi-hop traversal

Instruction / Prompt. Deterministic retrieval followed by a single LLM call.

1. **Query-focused subgraph.** The agent first builds a focused view of its domain that includes the domain label and description, a short summary of prior governance decisions for the domain, and the most query-relevant triples scored by term overlap with the question.
2. **Multi-hop expansion.** When the question mentions multiple entities present in the graph, the agent additionally retrieves shortest paths between them; when it mentions a single entity, it retrieves a small local neighbourhood of that entity. These paths are appended to the prompt as additional context.
3. **Answering.** System: *"You are a domain expert for '{domain.label}'. Answer queries using ONLY the knowledge graph data provided. Use multi-hop reasoning paths when available to explain indirect connections. Be precise about what you know and don't."* The user prompt embeds the focused subgraph and any multi-hop paths, the question itself, and a JSON return schema asking for `answer`, `coverage`, `evidence` (a list of cited triples in `(subject) -[relation]-> (object)` form), `confidence`, and `out_of_scope_aspects`. The prompt forbids mentioning domain IDs, the routing layer, or the phrase "knowledge graph," and forbids speculation beyond the supplied evidence.

Tools available.

- Domain access: `Domain.get_subgraph(...)` for owned entities and triples.
- Graph traversal helpers for shortest paths and entity neighbourhoods over the full graph.
- LLM: `chat_completion_json` for the single answering call.

Output. A per-domain answer record carrying `answer`, the cited `evidence` triples, an estimated `coverage` and `confidence`, any `out_of_scope_aspects` the agent could not address, and the originating domain identifier.

D.3 SynthesisAgent

Title: *SynthesisAgent*
Role Description. Combines per-domain expert responses and any cross-domain bridge context into a final user-facing answer with both an evidence-grounded prose response and a minimal short-form answer.
Inputs.
- `question: str`
- `domain_responses: List[Dict]` from each invoked `DomainExpertAgent`
- `cross_domain_context: str` — additional triples that bridge entities mentioned by different experts

Instruction / Prompt. Single LLM call. System: *"You are a knowledge synthesis expert. Combine partial answers into a coherent, evidence-grounded response. Return only valid JSON."* The user prompt lists, for each domain expert, its `answer`, cited `evidence`, and `out_of_scope_aspects`, followed by the bridge triples and the synthesis rules:
- *No meta-commentary.* The answer must not mention experts, routing, confidence scores, or the phrase "knowledge graph".
- *Evidence grounding.* Include only claims directly supported by the expert evidence or by triples in the cross-domain context. Both are treated as first-class evidence.
- *Multi-hop chaining.* For questions that span more than one fact, chain across triples: when an expert supplies a hop-1 entity and the cross-domain context contains a triple about that entity that resolves the next hop, use it rather than abstaining.
- *Relation paraphrasing.* Treat closely related relation labels as semantic equivalents when the question phrases them differently from the graph (e.g., a partnership relation can support a "spouse" or "collaborator" question; a "born in" relation can support a "birthplace" question).
- *Abstention.* If the supplied evidence does not support an answer, state the limitation briefly rather than speculating.
- *Two answer forms.* Return both an evidence-grounded `answer` (1–3 sentences) and a minimal `short_answer` (1–5 words; e.g., a name, a place, a year, "yes" / "no").

Tools available.
- LLM: `chat_completion_json` for the synthesis call.
- Cross-domain context assembly used to build the bridge-triple block in the prompt.

Output. A JSON record with the prose `answer`, the minimal `short_answer`, an overall `coverage` and `confidence`, and a list of remaining `gaps`. If no domain expert returned anything usable, the orchestrator emits a fixed abstention message instead of calling the LLM.

E Human Annotation Instructions

Annotation instructions:
1. Judge only from the evidence/source text shown.
2. Do not use outside knowledge.
3. If the triple is plausible but not directly supported, mark partial or no.
4. For usefulness, ask whether this triple would help KG search, QA, or reasoning.

TRIPLE: (iterative_deformation) –[Used-for]–> (reconstruction_process)

EVIDENCE / SOURCE TEXT:
iterative deformation of a 3 – D surface mesh to minimize an objective function We propose to incorporate a priori geometric constraints in a 3 – D stereo reconstruction scheme to cope with the many cases where image information alone is not sufficient to accurately recover 3 – D shape . Our approach is based on the iterative deformation of a 3 – D surface mesh to minimize an objective function . We show that combining anisotropic meshing with a non-quadratic approach to regularization enables us to obtain satisfactory reconstruction results using triangulations with few vertices . Structural or numerical constraints can then be added locally to the reconstruction process through a constrained optimization scheme . They improve the reconstruction results and enforce their consistency with a priori knowledge about object shape . The strong description and modeling properties of differential features make them useful tools that can be efficiently used as constraints for 3 – D reconstruction .

Q1. Does the source evidence support the triple?
1. yes
2. partial
3. no
4. unclear

Q2. Is the relation/predicate correct?
1. yes
2. partial
3. no

Q3. Are the subject and object/endpoints correct?
1. yes
2. partial
3. no
Q4. Is this triple useful for KG search, QA, or graph reasoning?
1. yes
2. maybe
3. no

Q5. Did the revision improve the triple?
1. improved
2. same
3. worse
4. unclear

Optional short note, or press Enter to skip: