Hybrid Knowledge Graph Construction via Multi-Level Chunking and Multi-Stage Quality Gating in Research Agentic AI MySRE

Muhammad Zaidan Rafat[1], Rio Nurtantyana[2], Ahmad Tabrani[3], Aria Bisri[4]

[1,2]Department Informatics, Faculty of Science & Technology, UIN Sultan Maulana Hasanuddin Banten, Serang, Indonesia
[2,4]Research Group HCI-V, Data & Information Science Center, National Research & Innovation Agency, Bandung, Indonesia

*Email corresponding author muhammadzaidanrafat@gmail.com

Abstract

Introduction

Scientific and engineering output has grown rapidly worldwide. The National Science Board (2024) reports that “From 2003 to 2022, annual S&E publications increased by roughly a third (36%) in the United States, whereas they increased approximately 10-fold in China and 8-fold in India,” while "Gold OA articles increased over 50-fold, from 19,000 articles published in 2003 to 992,000 articles in 2022" [1]. This volume of information creates what Lu et al. (2025) describe as "the exponential growth of scientific literature, with over 7 million articles published annually, which exposes a significant bottleneck: the widening gap between unstructured knowledge in texts and its structured representation in KGs" [2]. Consequently, "Traditional approaches to KG enrichment, such as manual curation, are reliable but unsustainable at scale" [2], requiring automated knowledge management systems.

Retrieval-Augmented Generation (RAG) emerged to handle such knowledge-intensive text. As Lewis et al. (2020) state, "Pre-trained neural language models cannot easily expand or revise their memory, can’t straightforwardly provide insight into their predictions, and may produce 'hallucinations'," thus “Hybrid models that combine parametric memory with non-parametric (i.e., retrieval-based) memories can address some of these issues” [3]. However, this basic flat vector approach suffers from context loss. Singh et al. (2026) clarify that "traditional RAG pipelines are typically static and linear, limiting complex multi-step reasoning, deep contextual understanding, and iterative response refinement" [4]. To solve this, the methodology shifts to Agentic RAG, which "integrates autonomous agents directly into the RAG pipeline to enable dynamic retrieval" [4]. For example, in the scientific domain, Lála et al. (2023) built PaperQA because "standard RAG models follow a fixed, linear flow," and they successfully "eliminate these limitations by breaking RAG into modular pieces, allowing an agent LLM to dynamically adjust and iteratively perform steps in response to the specific demands of each question" [5].

If an LLM builds a Knowledge Graph by pure brute-force calculation across documents ($O(N^2)$), it encounters fatal bottlenecks in computation and semantic integrity. Computationally, Huang et al. (2024) prove that “The fundamental issue is that LLMs cannot properly judge the correctness of their reasoning, LLMs struggle to self-correct their responses without external feedback” [6]. Without a Native Vector Index to prune the search space early, $O(N^2)$ scalability is impossible; Sehgal and Salihoğlu (2025) note that modern systems must use a “prefiltering approach” before connecting chunks with structured records like a knowledge graph [7]. Semantically, forcing LLMs to define relations without filters degrades accuracy. Berman et al. (2025) warn that “existing methodologies tend to overlook the generation of edge attributes,” and “Without edge features, this information would have to be redundantly stored at the node level, increasing complexity and obscuring transition flow” [8]. To prevent the graph from becoming a tangled 'hairball' of false relations, Hassan et al. (2025) assert that “Evaluation measures the quality of the KG, including its completeness, consistency, accuracy and relevance By evaluating the KG, errors and inconsistencies can be identified and corrected" [9].

This article proposes the Agentic Smart Research Environment (MySRE) as an architectural solution. MySRE introduces a hybrid extraction model using Multi-Level Chunking and Multi-Stage Quality Gating (K-NN Thresholding, Semantic Coherence, and Domain Compatibility) to bypass $O(N^2)$ complexity and filter semantic hallucinations. To demonstrate this superiority, generation quality is measured using Deepeval Framework. For the resulting Knowledge

Graph, topological metrics via NetworkX are calculated. Newman (2003) validates this by stating that graph theory "aims to find statistical properties, such as path lengths and degree distributions, that characterize the structure and behavior of networked systems" [10]. Finally, Seo et al. (2022) justify that "As a complement to the shortcomings of current structure-based metrics and data-quality-based assessment techniques, we introduce the structural quality metrics as a measure of knowledge graph quality" [11]. This ensures that MySRE produces structurally precise and highly organized results.

Methodology

2.1 MySRE v2.0 System Architecture Overview

MySRE v2.0 is organized as a two-stage architecture that transforms raw scientific PDFs into a knowledge graph. The first stage, Structural Node Generation, handles document preparation and pillar extraction. The second stage, Relational Edge Generation, builds validated relations from the resulting nodes. This separation keeps structural extraction, semantic cleanup, and relation building in distinct steps, so each stage works with a narrower input and a clearer output.

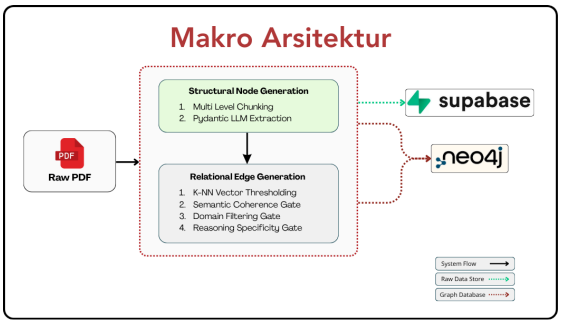

Figure 1. MySRE v2.0 system architecture overview, from raw PDF ingestion to node generation, gated edge construction, and storage in PostgreSQL and Neo4j.

In the first stage, the parser does not send raw PDF text directly to generation. It first converts the document into machine-readable text, removes retrieval noise, and groups the content into hierarchical chunks. A retrieval layer then selects the most relevant evidence for the five research pillars: background, methodology, objective, gap, and future work. That evidence is passed to the extractor so the final node stays tied to the source document rather than to isolated fragments.

The second stage creates graph relations only after the node has been accepted. Candidate edges are first screened by vector similarity, then filtered by semantic coherence, domain compatibility, and reasoning specificity. Only relations that survive every gate are materialized in Neo4j, while chunk records and ingestion metadata remain in PostgreSQL/Supabase. This arrangement keeps the graph compact and traceable because each stored edge is backed by a controlled evidence path rather than by a loose topical match.

2.2 *Structural Node Generation via Multi Level Chunking*

The pipeline begins by separating archival text from retrieval-ready text. Lewis et al. note that "ability to access and precisely manipulate knowledge is still limited" [3] , and Singh et al. add that “traditional RAG systems are constrained by static workflows” [4]. On long scholarly PDFs, that weakness appears as broken context, flat retrieval, and a loss of the document's internal order. The design therefore keeps the source text intact and sends only a cleaned, retrieval-ready view into node generation.

The parser emits two views of the same document. The archival view preserves the full source; the retrieval-ready view is stripped of reference lists, captions, and layout noise, then split by headers so the paper's hierarchy remains visible. PaperQA is relevant here because it "performs information retrieval across full-text scientific articles" [5]. Scholarly articles are not plain text dumps. They carry sections, sub-sections, and rhetorical roles, and the parser keeps that structure rather than flattening it.

Figure 2. Multi-level chunking pipeline for node generation.

Chunking on a fixed boundary is too coarse for scientific prose. RAPTOR states that "most existing methods retrieve only short contiguous chunks" [12], while HiChunk warns that "existing RAG evaluation benchmarks are inadequate for assessing document chunking quality" [13]. Dense X Retrieval sharpens the point further: "retrieval unit choice significantly impacts the performance of both retrieval and downstream tasks" [14]. The pipeline keeps that balance by using a parent-child layout, with parent chunks around 1,500 characters and child chunks around 240 characters. Child chunks carry retrieval, parent chunks carry context.

The two-level layout is not a cosmetic split. It is a retrieval strategy. Parent chunks keep the argumentative thread intact, while child chunks expose a smaller and more exact search surface. This is what allows the system to hold local evidence and global context at the same time. The child unit is small enough to search cleanly, but the parent unit is large enough to preserve the passage around the claim.

Retrieval is organized by rhetorical role. Background, methodology, gap, objective, and future work do not share the same surface form, so a single generic query is too blunt. The retrieval layer therefore builds a separate query for each pillar and routes it to the matching section family. Gap retrieval gets an extra pass: the system first works at the sentence level, then expands to the surrounding window when the first match is too thin. That keeps the search narrow without losing the line of reasoning that makes the passage useful.

The retrieval engine uses `intfloat/multilingual-e5-small`. Wang et al. describe text embeddings as "fundamental components in information retrieval systems" [15], which is the role they play here. The model gives the query and the child chunk a shared embedding space before similarity search starts. That matters in a corpus that mixes English and Indonesian technical text, because the embedding layer has to carry the query's intent before the retriever sees the passage.

The assembled context becomes the final input to the extractor. The extractor produces five pillar texts in a structured form, and those texts are written as a node in the graph database. Chunk records stay in a separate storage layer so retrieval and traceability remain available without bloating the graph. The split keeps the graph compact and the evidence path recoverable.

The extractor turns that context into five pillar texts and writes them as a graph node. Chunk records remain in separate storage so the evidence path stays traceable. The node is therefore not a free summary. It is a controlled unit of evidence, shaped by the document's hierarchy and by the retrieval path that found it.

2.3 Multi Stage Quality Gating

Traditional knowledge graph construction typically employs Cosine Similarity scores with static, hardcoded thresholds, commonly hitting flat marks such as ≥ 0.70. This blind approach ignores the inherent spatial biases of dense vectors, resulting in the Hairball Graph phenomenon where setting the threshold too low produces massive false connections, while setting it too high eradicates essential relations. MySRE introduces an Adaptive Statistical Threshold, mathematically calibrating the noise floor dynamically according to the actual data population distribution through the Three Sigma Rule of Thumb.

The system evaluates semantic distance by calculating the Cosine Similarity between document vectors.

$$\text{sim}(A, B) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \quad (1)$$

For any incoming query document q, the engine initially searches the database D to fetch a candidate set S bounded by a maximum limit top K=5. This hyperparameter strictly restricts the maximum candidate volume retrieved per query to prevent system memory overload (OOM):

$$S = \text{KNN}(q, D, K) \quad (2)$$

To dynamically identify anomalies, the system periodically samples a defined population of N=500 random pairs from the database during the distribution curve calibration process to compute the population mean (μ) and sample standard deviation (σ). An execution barrier is then constructed using the Z-Score multiplied by an academic pillar-specific sensitivity weight (M_{pilar}):

$$Z_{score} = \mu + 3\sigma \quad (3)$$

$$\text{Final Threshold}_{pilar} = Z_{score} \times M_{pilar} \quad (4)$$

Candidates within the set S must survive this brutal cutoff. The mathematical validity of this approach is founded upon the 3σ rule, which states that for any unimodal distribution, the probability of a random variable falling outside the interval $[\mu-3\sigma,\mu+3\sigma]$ is at most 5% and specifically 0.27% for an exact normal distribution [16]. Consequently, the system operates on the statistical anomaly detection principle, which assumes that normal data instances occur in high probability regions of a stochastic model, leading to the labeling of an observation as an anomaly if it possesses a low probability of being generated by that mode [17].

Table 1. Asymmetric Hyperparameter Weights (Multipliers) across Academic Pillars

Pillar	Value	Class
Background	1.0x	No penalty. Background sections are highly generalized and formulaic.
Methodology	1.0x	No penalty. Procedural and tooling methodologies are heavily repetitive
Objective	1.05x	Tightened by 5%. Research objectives must be sharp to be considered intersecting.
Gap	1.05x	Most Brutal Filtration. The gap determines research originality. The semantic vector must be nearly identical to form a relation (tightened by 15%).
Futureworks	1.15x	Tightened by 5%. Future projections require high lexical synchronization.

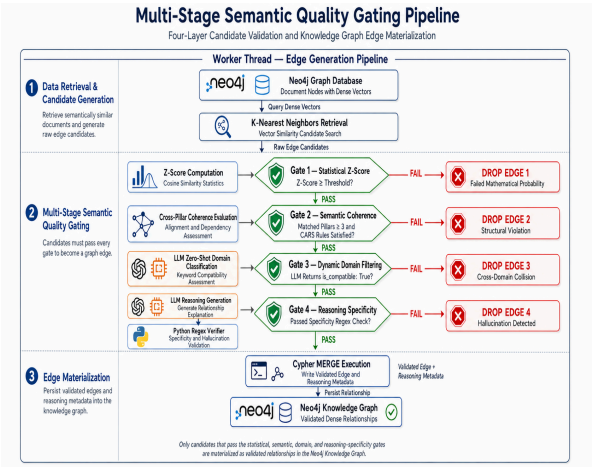

Fig. 4. Architecture of the Multi-Stage Quality Gating system, visualizing the sequence of the Statistical K-NN Vector Thresholding, Semantic Coherence Gate, Dynamic Domain Filtering Gate, and Reasoning Specificity Gate

Gate 2 - Semantic Coherence Gate:

Surviving the distance calculation in Gate 1 does not guarantee structural alignment. Traditional vector search architectures generate false positives by allowing relations based on single-pillar matches. For instance, two papers utilizing the Random Forest algorithm will register identical Methodology vectors, even if one predicts weather and the other forecasts heart failure. To counter this, MySRE implements the Semantic Coherence Gate, a verification layer enforcing cross-pillar validation to ensure relationships adhere strictly to academic rhetorical rules.

To maintain optimal execution speed without relying on the heavy O(N2) quadratic distance calculations required by optimal transport frameworks, MySRE reduces complexity through an absolute O(1) Majority Voting heuristic. The system parameter requires a minimum consensus of three match. Since the total ontology consists of five pillars, demanding three matches requires the consensus to exceed 50%. This cross-dimensional requirement reflects the empirical reality that scientific papers describe multifaceted arguments and ideas, suggesting that models capable of matching specific aspects can better capture overall document relatedness by flexibly aggregating over fine-grained sentence-level aspect matches [18].

Furthermore, these multi-aspect validations are bound by the CARS (Creating a Research Space) framework [19]. Under this rhetorical doctrine, the description of methodology acts solely as an operational instrument subordinate to the fulfillment of the research objective designed to resolve a specific gap. Translating this postulate into executed code, the system enforces a strict Boolean dependency matrix. A Methodology match is instantly nullified if it is not accompanied by an intersection in the Objective or Gap pillars, obliterating the formation of false algorithm clusters. Similarly, a shared Objective or Gap is deemed void without a corresponding procedural approach or defined research space.

Table 3.2: Boolean Dependency Matrix based on the CARS Framework

Anchor Pillar	Requirement	Ontological Enforcement Description
Methodology	Must be accompanied by **Objective** or **Gap**	Nullifies isolated methodology matches to eliminate the formation of false algorithm clusters
Objective	Must be accompanied by **Background**, **Methodology**, or **Gap**	Research objectives are merely verbs without supporting execution instruments (Methodology) or a localized problem (Gap/Background).
Gap	Must be accompanied by **Objective** or **Methodology**	Sharing a problem constraint is invalid unless both papers deploy a similarly structured execution or objective approach.
Futureworks	Must be accompanied by **Objective**, **Methodology**, or **Gap**	Future trajectory suggestions stem from current limitations; similarities are only valid if executed within allied problem perimeters.

Gate 3 – Dynamic Domain Filtering Gate:

Dense vectors evaluate text strictly via mathematical probability and spatial proximity, leaving them inherently blind to scientific taxonomy. Such cross-domain semantic collisions result in artificial cosine similarity spikes across completely divergent scientific disciplines. Instead of executing highly expensive computational fine-tuning on the base embedding model, MySRE shifts this validation burden to the generative Large Language Model, deploying it as a Zero-Shot Taxonomy Judge.

The generative model is restricted to analyzing only the specific research keywords extracted from the source and target nodes, rather than processing the entire document text. Driven by the explicit system prompt acting as an expert academic domain classifier, the model evaluates latent semantic meaning, normalizes multi-language taxonomy concepts, and resolves technical abbreviations. To protect backend automation from parsing errors, the model is strictly coerced to output a JSON object returning a pure Boolean `is_compatible` variable and a sub-ten-word `reason` string. A negative Boolean trigger results in the immediate truncation of the relational edge. This architectural choice is validated by literature

confirming that existing medical LLMs are typically either pre-trained or directly aligned to specialized domains by utilizing Zero-shot Prompting, where only task descriptions are allowed to be entered without past data examples [20].

Gate 4 – Reasoning Specificity Gate:

Large Language Models exhibit a severe vulnerability to sycophancy and epistemic blindness. As completion machines, these models possess a natural tendency to fabricate affirmative narratives rather than reject instructions when analyzing uncorrelated data [26]. Relying blindly on an LLM to formulate relational justifications would pollute the knowledge graph with generic, hallucinatory connections stating that the texts simply discuss similar topics.

MySRE counteracts this through the Reasoning Specificity Gate. The model is forced to execute a generation-time correction by producing a textual justification for the relationship. However, the system explicitly rejects the notion of LLM self-correction. Research unequivocally proves that LLMs struggle to self-correct their responses without external feedback, and their performance frequently degrades after self-correction attempts. Instead, the implementation utilizes a code executor to serve as the perfect verifier to judge correctness [6]. This external verification relies on deterministic Regular Expressions and keyword matching. Any reasoning text containing generic platitudes is subjected to a hard drop. Furthermore, the system implements an Atomic Facts doctrine, demanding the presence of specific operational passwords—such as dataset, metrics, or accuracy. Computing factual precision involves breaking generations into atomic facts containing isolated pieces of information, acknowledging the inherent trade-off between absolute precision and total recall [21]. Failure to manifest these concrete scientific instruments triggers the instant eradication of the candidate edge.

2.4 Synthetic Datasets

This synthetic reference dataset was prepared as the evaluation anchor for article-grounded testing. Following the benchmark logic described by Filice et al. [22]. it was organized as a controlled synthetic dataset rather than as a set of free-form prompts. Its structure contains two linked layers: a node reference for each source article and an edge reference for each document pair. The overall flow follows the document-to-context-to-generation sequence described in the DeepEval guides [23], [24], while retaining a quality-control step before the final artifacts are accepted [25].

A. Synthetic Node Reference

The synthetic node reference was generated from one source article and used as the reference target for node-level evaluation. It preserved a document-specific representation before any comparison with extracted graph nodes was performed. This direction follows the document-based synthesis pattern used in DeepEval, where test goldens are derived from supplied documents rather than from unconstrained prompts [23]. In this study, the same approach was applied to scientific articles so that each node reference remained tied to one source file.

Each PDF was first parsed into machine-readable text, then divided into smaller units for selection. The pipeline did not forward the entire document into generation. Instead, it filtered and assembled the selected material into a context bundle that supplied the evidence used at the next stage. DeepEval describes this flow as a context construction process in which documents are split into "smaller, manageable chunks," then selected and grouped for later generation . Its synthesis guide presents the same order at a higher level, moving from document loading and chunking to context generation [24] . The local implementation followed that sequence while applying stricter filtering to exclude weak or noisy spans.

The resulting context bundle was used to generate one synthetic node reference. The active model at this stage was `opus-4.7`, with `opus-4.6` serving as the fallback. The output remained constrained by the supplied context so that the reference reflected the source article itself. Jadon et al. capture this constraint through the phrase "context-anchored QA pairs" . That wording is relevant here because the node reference was intended to function as grounded evaluation data.

Each candidate then passed through quality control and structural validation before it was retained. AttackQA reports a similar practice, noting the use of `G-Eval` in DeepEval to support automatic curation through a quality control model [26]. In this pipeline, the critic stage served that filtering role. Weak candidates were rejected, and accepted outputs were checked against the required schema before being stored as final reference artifacts.

B. Synthhetic Edge Reference

The synthetic edge reference was generated for each document pair and used as the reference target for edge-level evaluation. This stage began only after valid node references had been established, so pair construction relied on curated document representations rather than on raw files [22]. In this

study, that staged design was used to control the relation-building process before any edge claim was produced.

Each document pair was then reduced to a compact evidence pack derived from the two node references. The generator compared both sides across background, methodology, gap, objective, and futurework using only the evidence retained in those references. As a result, the relation was inferred from paired evidence rather than from loose topical similarity. Jadon et al. describe a related constraint through the phrase "context-anchored QA pairs" [27], while the DeepEval guides place generation after document parsing, chunking, and context formation [23], [24]. The local edge stage inherited that earlier context work and condensed it into pairwise evidence for relation assessment.

Each generated edge was then normalized, validated against the required schema, and passed through a critic audit before it was retained. AttackQA reports the use of `G-Eval` in DeepEval to support automatic curation and to require a rationale for each score [26]. The same screening principle was applied here to examine positive relations before they were accepted. Weak positives were demoted, and only audited outputs that satisfied the schema were stored as final edge reference artifacts.

2.5 Evaluations

DeepEval is used as the LLM judge layer in this evaluation pipeline. The local orchestrator does not run it as a separate tool. It sends each system output into the correct mode, then pairs that output with the synthetic node or edge reference built earlier through [25], [28], [29].

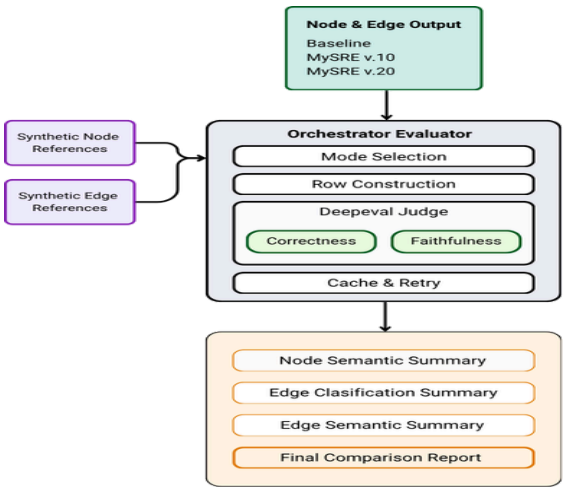

Figure X. Orchestrator evaluator pipeline, from mode selection to semantic scoring, edge classification, and report aggregation

The same judge layer is reused across node and edge modes. The orchestrator handles row construction, cache reuse, retry logic, and the final report.

A. Evaluation Orchestration

The evaluator starts from one entrypoint and branches into node, edge, and report modes. DeepEval serves as the judge layer in each mode. The orchestrator binds system outputs to the matching synthetic reference set before scoring begins.

1) Mode & Artifact Binding

Each run loads two inputs: the output from the system under test and the synthetic reference that acts as the gold target. Node and edge artifacts are stored in separate locations, so the orchestrator can bind each run to the correct reference set without cross-talk between modes.

2) Row construction and reuse

After loading, the orchestrator flattens each artifact into pillar-level rows. A node document yields five rows, one for each pillar. An edge pair yields the same five-row structure. Each row carries a stable input hash and a cache key. If a row has already been scored, the stored result is reused. If the judge call fails, the row is retried.

3) Report

Report mode gathers the node and edge outputs into summary matrices across systems. This keeps the comparison aligned across Baseline, MySRE v1.0, and MySRE v2.0, while preserving the same reference set and the same judge contract.

B. Node and Edge Evaluation Protocol

1) Unit evaluation

Node and edge cases are both scored at the pillar level. Two axes are used throughout: correctness and faithfulness. The split keeps semantic match separate from evidence support.

2) Node correctness

Node correctness compares the system output with the synthetic node reference. The judge looks for core meaning, not surface similarity. Omission, contradiction, and meaning shift count against the score. CondAmbigQA states that the outputs are evaluated using the "G-Eval function as implemented in the DeepEval package" [25], which matches the role of the correctness judge here.

3) Node faithfulness

Node faithfulness checks whether the system output is supported by the retrieved snippets. A topical match is not enough. Claims that reach beyond the snippets count as unsupported. LeMAJ is direct on the method used for this kind of scoring: "we use the G-Eval method of DeepEval" [28]. That line supports the use

of DeepEval as the judge layer for evidence-based grading.

4) Edge correctness

Edge correctness compares the binary edge decision and the relation reasoning against the synthetic edge reference. The judge checks the direction of the relation, its scope, and its pillar meaning. A loose semantic overlap is not enough. bLLeQA uses the same judge family when it says, "We leverage DeepEval's G-Eval metric" [29] , which makes it a close fit for the edge correctness rubric.

5) Edge faithfulness

Edge faithfulness checks whether the reasoning for a relation is grounded in the source and target snippets. Plausible relations that cannot be traced to evidence are treated as weak. The pairwise evidence view matters here because the edge is built from two documents, not one.

3. Scoring, Classification, and Report Aggregation

1) Pass/fail rule

A row passes only when both scores clear their thresholds:

$$\text{Pass} \iff (C \geq \theta_c) \wedge (F \geq \theta_f) \quad (5)$$

For node mode, θc =0.70 and θf=0.80 For edge mode, θc=0.80 and θf=0.80 The rule is strict. A row that misses either bound fails.

2) Semantic Summary

The evaluator then reduces the row scores into summary statistics:

$$\overline{C} = \frac{1}{N}\sum_{i=1}^{N} C_i \quad (6)$$

$$\overline{F} = \frac{1}{N}\sum_{i=1}^{N} F_i \quad (7)$$

$$\text{PassRate} = \frac{N_{\text{pass}}}{N} \quad (8)$$

These values are reported for node mode and for the semantic block of edge mode. They show how far each system stays inside the rubric, row by row.

3) Edge classification

Edge mode also produces a binary classification view. The four outcomes are TP, FP, FN, and TN.

Table 1. Edge Classification

Predicted Edge	Reference Edge	Class
1	1	TP
1	0	FP
0	1	FN

4) Report aggregation

Report mode gathers the node and edge outputs into summary matrices across systems. The final report carries a node semantic summary, an edge classification summary, and an edge semantic summary. It turns row-level scoring into a compact comparison of systems, pillars, and relation quality.

Result and Discussion

Conclusions

Reference

[1] N. C. for S. and E. Statistics (NCSES), "Publications Output: U.S. Trends and International Comparisons," Art. no. NSB-2023-33, Dec. 2023, Accessed: Jun. 16, 2026. [Online]. Available: https://ncses.nsf.gov/pubs/nsb202333

[2] Y. Lu, W. Wu, X. Zhao, R. Peng, and J. Wang, "KARMA: Leveraging Multi-Agent LLMs for Automated Knowledge Graph Enrichment," Jan. 11, 2026, *arXiv*: arXiv:2502.06472. doi: 10.48550/arXiv.2502.06472.

[3] P. Lewis *et al.*, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks", doi: https://doi.org/10.48550/arXiv.2005.11401.

[4] A. Singh, A. Ehtesham, S. Kumar, T. T. Khoei, and A. V. Vasilakos, "Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG," Apr. 01, 2026, *arXiv*: arXiv:2501.09136. doi: 10.48550/arXiv.2501.09136.

[5] J. Lála, O. O'Donoghue, A. Shtedritski, S. Cox, S. G. Rodriques, and A. D. White, "PaperQA: Retrieval-Augmented Generative Agent for Scientific Research," Dec. 14, 2023, *arXiv*: arXiv:2312.07559. doi: 10.48550/arXiv.2312.07559.

[6] J. Huang *et al.*, "Large Language Models Cannot Self-Correct Reasoning Yet," Mar. 14, 2024, *arXiv*: arXiv:2310.01798. doi: 10.48550/arXiv.2310.01798.

[7] G. Sehgal and S. Salihoglu, "NaviX: A Native Vector Index Design for Graph DBMSs With Robust Predicate-Agnostic Search Performance," Jun. 29, 2025, *arXiv*: arXiv:2506.23397. doi: 10.48550/arXiv.2506.23397.

[8] N. Berman and E. Kosman, "Reviving Life on the Edge: Joint Score-Based Graph Genera- tion of Rich Edge Attributes".

[9] H. Hassan, S. Elayidom, M. R. Irshad, and C. Chesneau, "A detailed analysis into evaluation metrics for knowledge graph evaluation," *Int. J. Data Sci. Anal.*, vol. 20, no. 8, pp. 6933–6972, Dec. 2025, doi: 10.1007/s41060-025-00871-3.

[10] M. E. J. Newman, "The structure and function of complex networks," *SIAM Rev.*, vol. 45, no. 2, pp. 167–256, Jan. 2003, doi: 10.1137/S003614450342480.

[11] S. Seo, H. Cheon, H. Kim, and D. Hyun, "Structural Quality Metrics to Evaluate Knowledge Graphs," Dec. 09, 2022, *arXiv*: arXiv:2211.10011. doi: 10.48550/arXiv.2211.10011.

[12] P. Sarthi, S. Abdullah, A. Tuli, S. Khanna, A. Goldie, and C. D. Manning, "RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval," Jan. 31, 2024, *arXiv*: arXiv:2401.18059. doi: 10.48550/arXiv.2401.18059.

[13] W. Lu, K. Chen, R. Qiao, and X. Sun, "HiChunk: Evaluating and Enhancing Retrieval-Augmented Generation with Hierarchical Chunking," Oct. 09, 2025, *arXiv*: arXiv:2509.11552. doi: 10.48550/arXiv.2509.11552.

[14] T. Chen *et al.*, "Dense X Retrieval: What Retrieval Granularity Should We Use?," Oct. 04, 2024, *arXiv*: arXiv:2312.06648. doi: 10.48550/arXiv.2312.06648.

[15] L. Wang, N. Yang, X. Huang, L. Yang, R. Majumder, and F. Wei, "Multilingual E5 Text Embeddings: A Technical Report," Feb. 08, 2024, *arXiv*: arXiv:2402.05672. doi: 10.48550/arXiv.2402.05672.

[16] F. Pukelsheim, "The Three Sigma Rule", [Online]. Available: https://www.jstor.org/stable/2684253

[17] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, Jul. 2009, doi: 10.1145/1541880.1541882.

[18] S. Mysore, A. Cohan, and T. Hope, "Multi-Vector Models with Textual Guidance for Fine-Grained Scientific Document Similarity," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Seattle, United States: Association for Computational Linguistics, 2022, pp. 4453–4470. doi: 10.18653/v1/2022.naacl-main.331.

[19] J. Swales, “Create a Research Space (CARS) Model of Research Introductions,” 1990.

[20] H. Zhou *et al.*, “A Survey of Large Language Models in Medicine: Progress, Application, and Challenge,” *arXiv.org*, 2023, doi: https://doi.org/10.48550/arXiv.2311.05112.

[21] S. Min *et al.*, “FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation,” Oct. 11, 2023, *arXiv*: arXiv:2305.14251. doi: 10.48550/arXiv.2305.14251.

[22] S. Filice, G. Horowitz, D. Carmel, Z. Karnin, L. Lewin-Eytan, and Y. Maarek, “Generating Q&A Benchmarks for RAG Evaluation in Enterprise Settings”.

[23] “Generate Goldens From Documents | DeepEval - The LLM Evaluation Framework.” Accessed: Jul. 23, 2026. [Online]. Available: https://deepeval.com/docs/synthesizer-generate-from-docs

[24] “Generate Synthetic Test Data for LLM Applications | DeepEval - The LLM Evaluation Framework,” DeepEval. Accessed: Jul. 23, 2026. [Online]. Available: https://deepeval.com

[25] Z. Li, Y. Li, H. Xie, and S. J. Qin, “CondAmbigQA: A Benchmark and Dataset for Conditional Ambiguous Question Answering”.

[26] V. B. Krishna, “AttackQA: Development and Adoption of a Dataset for Assisting Cybersecurity Operations using Fine-tuned and Open-Source LLMs,” Nov. 01, 2024, *arXiv*: arXiv:2411.01073. doi: 10.48550/arXiv.2411.01073.

[27] A. Jadon, A. Patil, and S. Kumar, “Enhancing Domain-Specific Retrieval-Augmented Generation: Synthetic Data Generation and Evaluation using Reasoning Models,” Feb. 21, 2025, *arXiv*: arXiv:2502.15854. doi: 10.48550/arXiv.2502.15854.

[28] J. Enguehard *et al.*, “LeMAJ (Legal LLM-as-a-Judge): Bridging Legal Reasoning and LLM Evaluation”.

[29] N. Banar, E. Lotfi, J. V. Nooten, M. Kliocaite, and W. Daelemans, “bLLeQA: Benchmarking LLMs for Grounded Legal Question-Answering in French and Dutch”.