KGAgent: A Continuously Updated Evidence Graph for Source-Traceable Medical Guide RAG

Shufeng Huang*†, Jihad Al Akl*‡, Qianxue Wang†, and Christophe Guyeux*
*Université Marie et Louis Pasteur, CNRS, institut FEMTO-ST (UMR 6174)
F-90000 Belfort, France
Email: shufeng.huang@univ-fcomte.fr
†School of Automation, Guangdong University of Technology
Guangzhou, China
‡Ticket Lab, Antonine University, Hadath-Baabda, Lebanon

***Abstract*—Medical regulation (dispatch) guides are highly structured documents. Answers must be traceable to a specific fiche and section, but conventional retrieval-augmented generation (RAG) often returns isolated text chunks and hides the document structure behind an answer. We present *KGAgent*, a filesystem-native GraphRAG prototype for local guide corpora. KGAgent converts directories, Markdown files, sections, keywords, references, and source metadata into a local knowledge graph without indexing-time LLM or embedding calls. The graph can be rebuilt, updated, or watched as local files are added, edited, or removed, so new knowledge becomes searchable without a remote database rebuild. KGAgent then performs graph-aware lexical retrieval over that graph. On 201 French medical regulation guide documents (3,695 nodes, 4,012 edges), we compare KGAgent with flat keyword, TF–IDF, dense-embedding, and two Neo4j GraphRAG baselines under the same corpus, query, and top-k protocol. We report bootstrap confidence intervals and paired significance tests. On the 31-query set, KGAgent attains Hit@5$=1.000$ and the highest context recall (0.879), while its rank precision is not significantly different from a strong TF–IDF baseline. On a 242-query expanded set, the split is clearer: TF–IDF significantly wins rank precision (Hit@1, MRR, nDCG@5), while KGAgent significantly wins context recall ($+0.047$, $p=0.002$). With a fixed generator (Mistral) and the same retrieval budget, KGAgent's retrieved context supports answers with a perfect source-hit rate and higher automatic quality than both Neo4j GraphRAG baselines. KGAgent also adds a pre-generation evidence check that refuses out-of-corpus questions before the language model is called. KGAgent's value is therefore not a blanket win over lexical retrieval, but *local, continuously updated, source-traceable evidence with explicit refusal*.**

***Index Terms*—GraphRAG, retrieval-augmented generation, knowledge graph, evidence traceability, source attribution, medical question answering.**

I. Introduction

Large language models (LLMs) increasingly serve as natural-language interfaces over specialized document collections, but they can produce fluent statements unsupported by evidence [1]. Retrieval-augmented generation (RAG) grounds generation in retrieved passages [2], [3] and is now common in knowledge-intensive and medical settings [4]. In safety-sensitive domains such as *medical regulation* (triage and dispatch advice for emergency call centers), grounding alone is not enough: an answer is useful only if it traces back to the exact guideline fiche and section, and the system should refuse when the corpus does not support the question.

French medical regulation guides make this concrete: not plain free text, each entry (a *fiche*) belongs to a clinical category, carries a record identifier and a source URL, and is organized into sections such as severity criteria, actions, equipment, keywords, and references. The corpus also changes over time; new fiches or procedure updates should become searchable as soon as files change, without rebuilding a remote database or rerunning an LLM/embedding indexing job. A chunk-based RAG pipeline retrieves useful text but usually loses the structure that tells a dispatcher *where* advice comes from and *how* sections relate. Knowledge graphs fit this structure, and recent GraphRAG surveys and systems make the direction explicit, from Microsoft GraphRAG [5] and LightRAG [6] to newer variants [7]–[9], though graph methods add real complexity and do not automatically beat strong vector or lexical retrieval [10]. Most existing systems still build an *LLM-extracted entity graph* offline and store it in an external database: useful for global summaries, but heavy for a local project that changes often.

We take a local-first position and introduce *KGAgent*, a *filesystem-native* GraphRAG prototype that builds a knowledge graph directly from the project's Markdown and source files using deterministic parsers, storing it as a single local JSON index that ordinary tools and coding agents can inspect, and retrieves over it with a graph-aware lexical scorer. The graph lives with the repository: when a file is added, edited, or removed, the build/update/watch loop synchronizes the corresponding nodes. Despite its name, KGAgent is not a new autonomous agent, but a graph-based evidence layer for agents and LLM applications, with no external database, embedding service, or LLM call in the indexing path, so it can be rebuilt or updated at zero external-service cost; model calls are needed only for a downstream answer. At query time, KGAgent asks the LLM to answer only from retrieved evidence with inline citations, and checks evidence strength first so weak out-of-corpus queries are refused early.

We do not claim that a structured graph always beats strong lexical retrieval: on a terminology-heavy guide corpus, TF–IDF is a very strong baseline. The contribution we defend is

more specific: KGAgent gives competitive retrieval while also returning *structured, source-traceable evidence* and an explicit refusal mechanism, two capabilities a flat retriever does not provide. Our contributions are:

- **A filesystem-native GraphRAG prototype** that converts a medical guide corpus into a local, self-updating knowledge graph of directories, documents, sections, keywords, references, and source metadata, keeps the index synchronized as local files change, and retrieves over it with an inverted-index, graph-aware lexical scorer without indexing-time LLM calls, embedding calls, or an external graph database (Section III).
- **A shared-protocol benchmark** on 201 French medical regulation guide documents against five baselines (flat keyword, TF–IDF, dense-embedding RAG, two Neo4j GraphRAG retrievers) under one corpus, query set, and top-k budget, with bootstrap confidence intervals and paired significance tests rather than bare point estimates (Sections IV–V).
- **A clear account of when graphs help.** With adequate statistical power (242 queries), lexical retrieval wins rank precision while KGAgent wins context recall; KGAgent's retrieved context also best supports a fixed downstream generator, among the GraphRAG systems compared, with perfect source-hit rate (Section V).
- **A pre-generation evidence-sufficiency check** that rejects questions when the retrieved graph evidence is too weak, preventing the language model from generating unsupported answers for out-of-corpus queries (Sections III–V).

KGAgent is a document-retrieval and evidence-navigation layer, not an autonomous clinical decision system. We state the limits of the current evidence in Section VI.

II. Related Work and Positioning

Retrieval-augmented generation. RAG combines an LLM with a retriever so that generation is grounded in external passages [2], and surveys show it has quickly grown into naive, advanced, and modular variants [3]. Retrieval is usually lexical (IDF [11], BM25 [12]) or dense (learned passage encoders [13] or sentence embeddings [14]). Long or poorly ordered context can reduce answer quality [15], which motivates retrieving concise, well-structured evidence rather than many loose chunks; recent work also asks whether retrieved context is sufficient, since insufficient context can lead models to answer incorrectly instead of abstaining [16]. KGAgent is lexical at its core but selects evidence through graph structure and section roles; we include both a TF–IDF and a dense baseline to isolate that effect.

GraphRAG and knowledge-graph-augmented retrieval. GraphRAG uses graph structure during indexing, retrieval, or generation; recent surveys describe it via graph-based indexing, graph-guided retrieval, and graph-enhanced generation [7], and broader roadmaps discuss combining knowledge graphs with LLMs [17]. Microsoft GraphRAG builds an LLM-extracted entity graph and precomputes community summaries for global, corpus-level questions [5]; LightRAG adds dual-level graph indexing and incremental update [6]; LinearRAG and TagRAG target cheaper construction or more scalable maintenance [8], [9]. This supports our motivation but also clarifies the difference: many GraphRAG systems still depend on entity extraction, embeddings, external graph/vector stores, or extra machinery, built over toolkits such as the Neo4j GraphRAG package [18] and LlamaIndex [19], and recent critiques show graph methods can be complex and should be compared with strong non-graph baselines [10]. KGAgent differs in two ways: its graph is *not* an LLM-extracted entity graph but a deterministic projection of the file system and document structure, so indexing needs no API calls and is cheap to keep in sync; and its index is a plain local JSON file rather than a database, so evidence and source information are directly inspectable. We benchmark against two Neo4j GraphRAG retrievers under the same protocol and include a dense baseline to avoid a weak comparison.

Medical question answering and faithfulness. LLMs encode clinical knowledge [20], and RAG is widely studied as a way to keep medical answers current and grounded [4], though a recent scoping review finds medical RAG still early, with many studies relying on public datasets, English-centric embedding models, generic LLMs, and limited safety evaluation [21]. Because errors are costly, faithfulness and attribution are central: surveys document hallucination in generation [1], and benchmarks such as ALCE evaluate whether models cite supporting evidence [22]. KGAgent targets this need in a French medical-regulation corpus, returning section-level source information, supporting local knowledge updates, and adding an explicit insufficient-evidence refusal; we evaluate these behaviors with source-hit rate, citation validity, and negative out-of-corpus queries.

Positioning of our contribution. Compared with prior GraphRAG, KGAgent replaces global LLM-extracted entity reasoning with a *local, deterministic, filesystem-native* graph aimed at source-traceable retrieval over structured document corpora; compared with efficient GraphRAG variants, it uses the existing file hierarchy and Markdown structure as the graph substrate, making updates cheap and inspectable rather than learned or service-dependent; compared with lexical and dense RAG, it adds structure-aware evidence selection and refusal without an external index. Our claims are narrow and testable: KGAgent is competitive on retrieval but *not* a blanket winner over TF–IDF; its measured benefits are higher context recall, perfect source grounding for a fixed generator, and a calibrated refusal check (Section V).

III. Proposed Solution: KGAgent

Fig. 1 summarizes KGAgent in three stages: continuous local update, deterministic offline indexing, and retrieval with answer generation. Local file changes trigger graph refreshes; guide records are converted into Markdown fiches and projected into a local graph; at query time, graph-aware retrieval

uses the graph to support cited answers or insufficient-evidence refusals.

Corpus conversion. The corpus is generated from the public French *Guide d'aide à la régulation médicale*.[1] A deterministic builder fetches each record and writes one Markdown document per fiche, preserving the record identifier, clinical category, source URL, title, and update metadata, so every retrieved result traces back to its origin. The resulting corpus contains 201 documents across 23 categories (Table I).

Graph construction. KGAgent walks the corpus directory and projects it into a typed graph $G = (V, E)$ using deterministic parsers only, with no embeddings or LLM. Node types include Project, Directory, MarkdownFile, Section, Concept, Keyword, Reference, and, for code files, Symbol/Dependency. Edges encode filesystem containment (contains), document structure (has_section), lexical/metadata relations (mentions, has_keyword, has_reference, links_to), and document-family relations (has_detail/summarized_by) inferred from fiche-code patterns. Each Section node stores its heading, body, level, source path, source title, and *section role*; roles such as *actions*, *risks*, *equipment*, and *references*, derived from heading text, help the retriever prefer evidence matching the query intent. For the study corpus, the graph has 3,695 nodes and 4,012 edges (Table I).

Continuous local knowledge update. KGAgent is a living local knowledge layer, not a one-shot index. The graph is saved as a single `graph.json` file and refreshed as the local file library changes: new Markdown files become new document/section nodes, edited files are re-parsed in place, deleted files have their subgraphs removed, and a polling watcher can keep the index synchronized during active editing. This matters for project knowledge bases where users keep adding fiches, notes, or procedures: the retrieval layer should expose that knowledge without rebuilding a remote database or rerunning an LLM extraction pipeline, and the update path being parser-based means no LLM extraction, embedding recomputation, or remote database synchronization.

Graph-aware retrieval. Given a query, KGAgent scores nodes with an IDF-weighted lexical model over several node fields with different weights: *name* (4.0), structured *metadata* (3.2), *path* (2.6), *text* (2.3), and an *identity* field (0.8). Per-term scores use a BM25-style saturation $\mathrm{tf} \cdot 2.2/(\mathrm{tf} + 1.2)$ weighted by idf [11], [12], with a bonus for exact phrase matches and a query-coverage multiplier for nodes covering more query terms. Structure-aware bonuses promote Section nodes and *actions*-role sections for procedural queries, while Keyword/Reference nodes are down-weighted. To keep latency low, candidate nodes are recalled through an inverted index over token variants, and graph-neighbor lookups are cached. An optional reasoning mode expands seed nodes along weighted edges to rank a local subgraph and expose evidence paths. Retrieval returns the top-k nodes with source paths and neighbors: KGAgent returns evidence with structure, not only text spans.

[1] https://www.guide-regulation-medicale.fr/fr/

TABLE I
CORPUS AND GRAPH STATISTICS.

Item	Value
Markdown documents	201
Clinical categories	23
Markdown characters (approx.)	1,036,611
Graph nodes	3,695
Graph edges	4,012
JSON index size (bytes)	4,638,216
Base queries (positive / negative)	31 (29 / 2)
Expanded queries (positive / negative)	242 (230 / 12)
Top-k retrieved	5

Evidence-sufficiency check. Before any LLM call, KGAgent computes a confidence score from the retrieved evidence,

$$\mathrm{conf} \;=\; s_{\text{top}} \cdot \big(0.5 + 0.5 \cdot c\big), \tag{1}$$

where s_{top} is the top retrieval score and $c \in [0, 1]$ is the fraction of query terms covered by the retrieved context. If conf falls below a method-specific threshold, the system refuses with "insufficient evidence" rather than prompting the model; the threshold is calibrated per method by maximizing balanced accuracy over positive queries and deterministic negative stress queries. When evidence is sufficient, the LLM is prompted to answer *only* from the retrieved context and cite source markers (`[1]`, `[2]`, ...), and to say so if the context cannot answer: a retrieval-level check combined with a prompt-level constraint, instead of relying on the prompt alone.

IV. EXPERIMENTAL SETUP

Corpus and graph. Table I reports corpus and graph statistics. All methods retrieve over a single record set derived from the 201 documents.

Query sets. The base set has 31 source-labeled queries: 29 positive with a gold source fiche and 2 negative out-of-corpus questions, covering exact-title lookup, keyword retrieval, triage priority, urgency levels, organization/process fiches, and phone guidance. For statistical power, we also build a 242-query expanded set (230 positive, 12 negative): the 29 original queries plus 201 template-derived questions across all 23 categories and five phrasing styles (exact title, keyword, paraphrase, caller-style, noisy), with every gold source and title key-point verified. As noted in Section VI, the derived queries reuse fiche terminology; they tighten confidence intervals but do not replace human evaluation.

Baselines. We compare six retrievers under the same top-k=5 budget (Table II). Lexical/dense baselines run over the same record set as KGAgent; the two Neo4j GraphRAG retrievers use a local multilingual MiniLM encoder [14] for seed retrieval and Neo4j for graph/Cypher expansion [18].

Metrics and statistics. We report Hit@1, Hit@5, MRR, nDCG@5 [23], and *context recall* (fraction of gold key-points present in the retrieved context), computed over positive queries (n=29 base, n=230 expanded) since negatives have no gold source. We report percentile bootstrap 95% CIs with 10,000 resamples [24], and paired tests against KGAgent: exact McNemar for binary metrics [25], two-sided paired bootstrap for continuous ones.

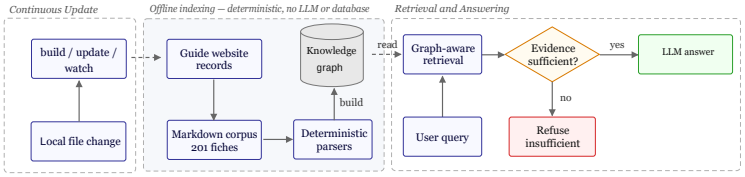


Fig. 1. End-to-end KGAgent workflow: continuous local update (`build`/`update`/`watch`), deterministic offline indexing with no LLM or external database, and retrieval with cited answers or evidence-insufficiency refusal.

TABLE II
RETRIEVAL METHODS UNDER THE SHARED PROTOCOL.

Method	Description
`flat_keyword`	Deterministic keyword-overlap baseline.
`tfidf_rag`	Deterministic TF–IDF lexical-vector baseline.
`dense_rag`	Dense embedding RAG (MiniLM, cosine); no graph, no Neo4j.
`kgagent_full`	KGAgent graph-aware lexical retrieval (this work).
`neo4j_driver_graphrag`	MiniLM seed retrieval + Neo4j graph expansion.
Neo4j official VectorCypher	Official `neo4j-graphrag` VectorCypher retriever over a Neo4j vector index.

TABLE III
RETRIEVAL EFFECTIVENESS ON THE BASE SET (n=29 POSITIVE QUERIES, TOP-k=5). BEST PER COLUMN IN **BOLD**.

Method	H@1	H@5	MRR	nDCG@5	C.Rec.
`dense_rag`	0.517	0.690	0.588	0.603	0.460
`flat_keyword`	0.759	0.897	0.821	0.817	0.831
`tfidf_rag`	**0.966**	**1.000**	**0.983**	**0.981**	0.798
`kgagent_full`	0.897	**1.000**	0.921	0.935	**0.879**
`neo4j_driver`	0.517	0.828	0.656	0.679	0.637
`neo4j_official`	0.483	0.655	0.552	0.562	0.395

Answer-generation protocol. Each method's top-k evidence is passed to an LLM with the same citation-and-refusal prompt. To avoid a model confound in the strongest comparison, KGAgent and the two Neo4j baselines use one fixed hosted model, Mistral `mistral-small-latest` [26].[2] We also report a smaller diagnostic over local methods using Gemini [27], read with that caveat. We score answer accuracy, automatic quality, key-point recall, source-hit rate, and citation validity.

V. RESULTS

Retrieval effectiveness. Table III shows the base-set results. KGAgent reaches Hit@5=1.000 and the highest context recall (0.879); TF–IDF is strongest on rank precision; both Neo4j baselines and the dense baseline trail. This suggests plain dense embedding retrieval, not only the Neo4j configuration, is the weak link on this terminology-heavy French corpus: since the dense and both Neo4j baselines share a multilingual MiniLM encoder, part of the gap reflects that encoder's weakness on French clinical terms rather than the graph design, so those numbers should be read as encoder-limited.

At n=29, a single query changes any Hit@k rate by about 0.034, so intervals are wide: KGAgent has Hit@1=0.897 with 95% CI $[0.759, 1.000]$, Hit@5 CI $[1.000, 1.000]$, and context recall 0.879 with CI $[0.815, 0.936]$. Paired tests show KGAgent significantly outperforms the dense and both Neo4j baselines on most metrics ($p \leq 0.004$), but versus TF–IDF is *not* significant on any metric (Hit@1 p=0.625, Hit@5 p=1.000, MRR p=0.179, nDCG p=0.190, context recall p=0.132): at this sample size the two methods are statistically indistinguishable, and we avoid any "win" claim over TF–IDF on the base set.

[2]Mistral AI API; model documentation at https://docs.mistral.ai.

TABLE IV
RETRIEVAL ON THE EXPANDED SET (n=230 POSITIVE; NEO4J METHODS REQUIRE THEIR SERVER AND ARE OMITTED). Δ IS KGAGENT−TF–IDF WITH THE PAIRED p-VALUE, COMPUTED ON UNROUNDED VALUES (MAY DIFFER FROM ROUNDED CELL DIFFERENCES BY UP TO 0.001).

Method	H@1	H@5	MRR	nDCG@5	C.Rec.
`flat_keyword`	0.630	0.878	0.741	0.783	0.536
`dense_rag`	0.744	0.839	0.784	0.788	0.493
`tfidf_rag`	**0.996**	**1.000**	**0.998**	**0.996**	0.842
`kgagent_full`	0.965	0.996	0.974	0.974	**0.888**
Δ (KGA−TFIDF)	−.030	−.004	−.024	−.021	+.047
p-value	.039	1.000	.012	.006	.002

On the 242-query set (n=230 positive), intervals shrink by roughly a factor of five and the KGAgent–TF–IDF relationship becomes a clear and significant split (Table IV). TF–IDF significantly wins rank precision: Hit@1 (−0.030, p=0.039), MRR (−0.024, p=0.012), nDCG@5 (−0.021, p=0.006). KGAgent significantly wins context recall (+0.047, p=0.002), while Hit@5 is tied (p=1.000), stronger than the base-set "perfect Hit@5" framing. Lexical retrieval is better at placing the single correct fiche first, while the graph retrieves more supporting evidence. We did not apply a multiple-comparison correction; under a Holm correction across the five metrics, the context-recall, nDCG@5, and MRR differences remain significant, while the borderline Hit@1 difference (p=0.039) does not, so we read that gap with caution.

Graph ablation. Removing graph signals one at a time leaves most metrics unchanged (Table V). The main exception is structure metadata (titles, section roles, file paths): removing it lowers Hit@1 from 0.897 to 0.862, Hit@5 from 1.000 to 0.966, and context recall from 0.879 to 0.847. On this corpus, structured document metadata helps ranking more than keyword or reference nodes.

Efficiency. Table VI reports median query latency. KGAgent is slower than the lexical baselines, but after inverted-index candidate recall and neighbor caching its median query time dropped from about 1 s in an earlier unoptimized run to 194 ms, with *no* change in retrieval metrics, on par with the Neo4j driver baseline and well below the one-time index-build

TABLE V
KGAGENT GRAPH ABLATION (n=29).

Variant	H@1	H@5	nDCG@5	C.Rec.
full	0.897	1.000	0.935	0.879
no_keywords	0.897	1.000	0.935	0.879
no_references	0.897	1.000	0.935	0.879
no_family_edges	0.897	1.000	0.935	0.879
no_structure_metadata	0.862	0.966	0.909	0.847

TABLE VI
MEDIAN QUERY LATENCY (TOP-k=5). BUILD/UPDATE ARE ONE-TIME KGAGENT INDEXING COSTS.

Operation / Method	Latency (ms)
`flat_keyword` (median query)	12.9
`tfidf_rag` (median query)	41.4
`neo4j_official` (median query)	135.6
`neo4j_driver` (median query)	194.3
`kgagent_full` (median query)	194.3
KGAgent full build	2,345.0
KGAgent no-change update	523.8

costs (2,345 ms full build, 524 ms no-change update). The trade-off is clear: structured, traceable evidence costs more per query than a flat lexical scan.

This local compute cost is distinct from external service cost: KGAgent is not cost-free in the physical sense, still consuming CPU time and local storage (single `graph.json` index, 4.64 MB for this corpus). Its cost advantage is that indexing and update require zero LLM calls, zero embedding calls, and no external graph/vector database or remote API.

Answer generation under a fixed model. Table VII compares KGAgent with the two Neo4j GraphRAG baselines under one generator (Mistral) and the same retrieval budget: KGAgent-retrieved context gives the highest answer accuracy (1.000), automatic quality (96.1), key-point recall (0.911), a perfect source-hit rate, and high citation validity (0.936), in a same-corpus, same-query, same-top-k, same-model comparison against implemented external GraphRAG baselines. A separate diagnostic over the local retrievers uses Gemini, subject to API quota limits: flat keyword, TF–IDF, and KGAgent all reach 0.968 answer accuracy, with quality scores of 92.2, 94.6, and 93.5. This matches the retrieval analysis: TF–IDF is competitive on automatic quality, while KGAgent attains a perfect (1.000) source-hit rate, so we attribute KGAgent's answer-generation advantage to *source grounding*, not to a blanket quality win.

Evidence-sufficiency check. Table VIII reports the calibrated refusal check from Eq. (1). For KGAgent, the calibrated threshold gives 0.951 accuracy, a 0.966 positive accept rate, and a 0.917 negative reject rate: it accepts answerable questions while rejecting most out-of-corpus questions *before* the LLM is called. TF–IDF calibrates perfectly on this small negative set. Because each threshold is selected by maximizing balanced accuracy on the same pool it is then scored on, these numbers are in-sample and should be read as an optimistic upper bound; a held-out calibration/test split remains future work. The negative pool is small (2 original negatives plus 10 deterministic stress negatives), so we treat the check as a working mechanism, not a validated clinical refusal system.

TABLE VII
ANSWER GENERATION UNDER ONE FIXED MODEL (MISTRAL), SAME PROTOCOL (n=31). ACC. = ANSWER ACCURACY; QUAL. = AUTOMATIC QUALITY (0–100); KPR = KEY-POINT RECALL; S-HIT = SOURCE-HIT RATE; CITE = CITATION VALIDITY.

Method	Acc.	Qual.	KPR	S-Hit	Cite
`kgagent_full`	**1.000**	**96.1**	**0.911**	**1.000**	**0.936**
`neo4j_driver`	0.774	80.8	0.750	0.828	0.774
`neo4j_official`	0.742	69.6	0.637	0.655	0.677

TABLE VIII
CALIBRATED EVIDENCE-SUFFICIENCY CHECK. PAR = POSITIVE ACCEPT RATE; NRR = NEGATIVE REJECT RATE.

Method	Thresh.	Acc.	PAR	NRR
`flat_keyword`	4.383	0.878	0.931	0.750
`kgagent_full`	73.906	0.951	0.966	0.917
`neo4j_driver`	0.467	0.902	0.862	1.000
`neo4j_official`	0.483	0.805	0.759	0.917
`tfidf_rag`	0.232	1.000	1.000	1.000

Robustness and agent utility. On the first ten positive queries, KGAgent preserves Hit@5=1.0 on the original phrasing and 0.9 under paraphrase and typo perturbations. In a corpus-growth stress test, Hit@5 stays at 1.0 at $1\times$, $2\times$, and $4\times$ duplicated scale, while median latency rises from 234 ms to 765 ms (these stress-test latencies run on a query subset, not directly comparable to the optimized full-set median in Table VI). In a scripted agent-utility test, a KGAgent query policy reaches a 1.0 task-success rate and a 1.0 evidence-trace rate, versus only 0.5 evidence-trace and three orders of magnitude more tool calls for a manual-scan policy, suggesting graph access helps agents navigate the corpus with reliable source traces.

Qualitative cases. Three cases illustrate the behavior behind the metrics. (i) *Purpura* (`grm_q007`): flat keyword misses the fiche, KGAgent recovers it at rank 4, TF–IDF ranks it first: a recovery case that also exposes a ranking weakness. (ii) *Anxiety crisis* (`grm_q017`): TF–IDF retrieves the correct fiche but thinner context, while KGAgent also retrieves reasoning and severity-analysis sections, raising key-point coverage. (iii) *Out-of-corpus* (`grm_q030`, a "lunar-hospital teleportation" question): both generators refuse rather than inventing a procedure. Together, these cases show where KGAgent is most useful: answers needing structured sections, source paths, and refusal, rather than a single exact title.

VI. LIMITATIONS

Our evidence has clear boundaries. (1) The query sets favor lexical methods: the base set is terminology-heavy, and the expanded set is derived from fiche text, so neither replaces independently authored, caller-style questions. (2) Automatic answer-quality and key-point metrics are proxies, not clinical safety judgments; we claim no medical correctness beyond document-grounded citation behavior, and KGAgent is not an autonomous clinical system. (3) The refusal check is calibrated on a small negative pool and should be re-validated with realistic near-miss medical negatives. (4) The fixed-model answer-generation comparison covers KGAgent and the two Neo4j baselines; extending it to the lexical and dense retrievers

is left to future work, so our generation claims are scoped to the GraphRAG systems compared. (5) We benchmark two Neo4j GraphRAG retrievers under a shared protocol; Microsoft GraphRAG [5], LightRAG [6] (adapter implemented, run pending), and LlamaIndex property graphs [19] are not yet run end-to-end, so we claim no superiority over all GraphRAG systems. (6) The methods do not all retrieve at the same granularity: KGAgent ranks section-level nodes, while lexical/dense baselines rank document/record units, so the shared top-k budget is not strictly unit-matched and the context-recall comparison should be read with this caveat. (7) The continuous-update loop is demonstrated functionally and stress-tested for corpus growth, but update latency or correctness on real edit streams is not evaluated.

VII. Conclusion

We presented KGAgent, a filesystem-native GraphRAG prototype for structured medical regulation guides, turning a local guide corpus into a self-updating, inspectable knowledge graph and retrieving source-traceable evidence over it. Under a shared protocol with bootstrap CIs and paired tests, KGAgent is competitive with a strong TF–IDF baseline: statistically tied at small scale, trading rank precision for significantly higher context recall with more queries. It also outperforms dense and Neo4j GraphRAG baselines and gives a fixed downstream generator perfect source grounding. Its main value is structured evidence with section-level source information, citation-aware answering, and a calibrated pre-generation refusal check, at an explicit latency cost. Future work: independently authored caller-style and multilingual queries, expert faithfulness review, single-model and additional GraphRAG baselines under the same protocol, and a refusal check improved with realistic near-miss negatives.

Acknowledgement

The authors thank Guangdong University of Technology and the FEMTO-ST Institute (UMR 6174 CNRS, Université Marie & Louis Pasteur) for funding and computing resources.

References

[1] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," *ACM computing surveys*, vol. 55, no. 12, pp. 1–38, 2023.

[2] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in neural information processing systems*, vol. 33, pp. 9459–9474, 2020.

[3] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, H. Wang *et al.*, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, 2023.

[4] G. Xiong, Q. Jin, Z. Lu, and A. Zhang, "Benchmarking retrieval-augmented generation for medicine," in *Findings of the Association for Computational Linguistics: ACL 2024*, 2024, pp. 6233–6251.

[5] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, D. Metropolitansky, R. O. Ness, and J. Larson, "From local to global: A graph rag approach to query-focused summarization," *arXiv preprint arXiv:2404.16130*, 2024.

[6] Z. Guo, L. Xia, Y. Yu, T. Ao, and C. Huang, "Lightrag: Simple and fast retrieval-augmented generation," *arXiv preprint arXiv:2410.05779*, 2024.

[7] B. Peng, Y. Zhu, Y. Liu, X. Bo, H. Shi, C. Hong, Y. Zhang, and S. Tang, "Graph retrieval-augmented generation: A survey," *ACM Transactions on Information Systems*, vol. 44, no. 2, pp. 1–52, 2025.

[8] L. Zhuang, S. Chen, Y. Xiao, H. Zhou, Y. Zhang, H. Chen, Q. Zhang, and X. Huang, "Linearrag: Linear graph retrieval augmented generation on large-scale corpora," *arXiv preprint arXiv:2510.10114*, 2025.

[9] W. Tao, X. Li, Y. Lan, and W. Qian, "Tagrag: Tag-guided hierarchical knowledge graph retrieval-augmented generation," *arXiv preprint arXiv:2601.05254*, 2026.

[10] R. Tuora, M. Galiński, M. Godziszewski, M. Karpowicz, M. Czyżnikiewicz, A. Kozakiewicz, and T. Ziętkiewicz, "Unweaving the knots of graphrag–turns out vectorrag is almost enough," *arXiv preprint arXiv:2603.29875*, 2026.

[11] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[12] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: BM25 and beyond," *Foundations and Trends in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.

[13] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," in *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, 2020, pp. 6769–6781.

[14] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, 2019, pp. 3982–3992.

[15] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the middle: How language models use long contexts," *Transactions of the association for computational linguistics*, vol. 12, pp. 157–173, 2024.

[16] H. Joren, J. Zhang, C.-S. Ferng, D.-C. Juan, A. Taly, and C. Rashtchian, "Sufficient context: A new lens on retrieval augmented generation systems," in *International Conference on Learning Representations (ICLR)*, 2025.

[17] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, "Unifying large language models and knowledge graphs: A roadmap," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 7, pp. 3580–3599, 2024.

[18] Neo4j, Inc., "The Neo4j GraphRAG package for Python," https://neo4j.com/docs/neo4j-graphrag-python/, 2024, software library; accessed 2026-06.

[19] J. Liu, "LlamaIndex: A data framework for LLM applications," https://github.com/run-llama/llama_index, 2022, software framework; accessed 2026-06.

[20] K. Singhal, S. Azizi, T. Tu, S. S. Mahdavi, J. Wei, H. W. Chung, N. Scales, A. Tanwani, H. Cole-Lewis, S. Pfohl *et al.*, "Large language models encode clinical knowledge," *Nature*, vol. 620, no. 7972, pp. 172–180, 2023.

[21] R. Yang, M. Y. H. Wong, H. Li, X. Li, W. Zhu, J. Liao, K. Yu, J. C. K. Liew, W. Xuan, Y. Chen *et al.*, "Retrieval-augmented generation in medicine: A scoping review of technical implementations, clinical applications, and ethical considerations," *arXiv preprint arXiv:2511.05901*, 2025.

[22] T. Gao, H. Yen, J. Yu, and D. Chen, "Enabling large language models to generate text with citations," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 6465–6488.

[23] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 422–446, 2002.

[24] B. Efron, "Bootstrap methods: another look at the jackknife," *The Annals of Statistics*, vol. 7, no. 1, pp. 1–26, 1979.

[25] Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," *Psychometrika*, vol. 12, no. 2, pp. 153–157, 1947.

[26] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier *et al.*, "Mistral 7B," *arXiv preprint arXiv:2310.06825*, 2023.

[27] G. Team, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican *et al.*, "Gemini: a family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, 2023.