Training-Free Hybrid Evidence Retrieval for Question Answering: Dynamic Fusion of Knowledge-Graph Triples and Dense Text Embeddings

Otávio Calaça Xavier[1], **Anderson da Silva Soares**[1]

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Alameda Palmeiras, Quadra D, Câmpus Samambaia – 74690-900 – Goiânia – GO – Brazil

{otaviocx,andersonsoares}@ufg.br

Abstract. *This work presents a sub-50ms, training-free retrieval pipeline that leverages a Neo4j knowledge graph and a ChromaDB vector index. Questions and passages are embedded with Sentence-BERT, and the retrieved entities seed a one-hop Cypher expansion in the knowledge graph. A transparent fusion based on Dice-Sørensen overlap ranks both passages and triples. On the WebQSP and CQA-12k benchmarks, this hybrid method achieves superior Recall@10, MRR, and nDCG@10 compared to BM25, graph-only, and vector-only baselines. Requiring no learned parameters and running on commodity hardware, it offers a practical alternative to heavyweight neural re-rankers and a robust evidence layer for retrieval-augmented generation (RAG).*

1. Introduction

Open-domain question-answering systems often rely on retrieving external evidence to answer questions. Two primary sources of knowledge are commonly used: structured knowledge bases (KBs) and unstructured text corpora. Each has complementary strengths and limitations. Knowledge graphs (such as Wikidata or Freebase) store facts in a structured form (subject–predicate–object triples) enabling precise querying, but are usually incomplete or may lack contextual descriptions. However, large text collections (e.g., Wikipedia) provide rich contextual information and coverage, but retrieving facts from text can be less precise and may require more complex reasoning. Recent research in open-domain QA has explored the combination of these sources to leverage their complementary advantages [Sun et al. 2019]. For example, early fusion methods such as GRAFT-Net integrated text and KB data in a single graph structure and achieved substantially better accuracy than using either source alone [Sun et al. 2018].

More recent hybrid retrieval-augmented generation approaches have also shown that using both a vector-based text retriever and a knowledge graph can outperform text-only or graph-only systems in both retrieval accuracy and downstream answer quality [Sarmah et al. 2024]. These works suggest that a hybrid approach can improve coverage of relevant facts and overall QA performance, especially when the knowledge graph is incomplete and questions may require additional textual context [Sun et al. 2019].

This paper proposes a novel hybrid evidence retrieval pipeline for question answering that dynamically combines evidence from a knowledge graph and a text corpus. In contrast to previous hybrid QA systems that often require complex neural architectures or joint training (e.g., graph neural networks for multi-hop reasoning [Sun et al. 2019] or

unified indexing of triples and text [Ju et al. 2022]), our approach emphasizes a simpler, modular design. We index structured facts in a Neo4j graph database and unstructured text in a vector database separately, then perform retrieval on both in parallel. One of the main contributions of this work is a fusion step that dynamically ranks evidence based on lexical similarity to the query (it uses the Dice–Sørensen coefficient) to merge and rank the results. This fusion method is lightweight and unsupervised, yet effectively filters redundant information and balances the contributions of structured and unstructured sources. It provides a simple but effective means to exploit the precision of knowledge graphs together with the broad coverage of text retrieval.

The hybrid retrieval system proposed in this paper is evaluated on two QA datasets. WebQuestionsSP (WebQSP) and CQA-12k. WebQSP consists of factual questions answerable using a knowledge base (originally Freebase) with 4,700 questions widely used in knowledge base QA research[1]. CQA-12k is a subset of the Complex Sequential Question Answering (CSQA) dataset [Saha et al. 2018], containing 12,000 independent QA pairs across various types of questions (simple, logical, quantitative, etc.), derived from Wikidata. These datasets allow us to assess the retriever's ability to find relevant evidence for different complexity levels and knowledge sources. The evaluation focuses on retrieval performance metrics, in particular Recall@10 (did the top 10 retrieved items include the evidence bearing the answers?), Mean Reciprocal Rank (MRR) of the first relevant result and nDCG@10 (a graded ranking quality metric).

The results indicate that the proposed hybrid approach consistently improves Recall@10 over using only the knowledge graph or only text. For example, on WebQSP our hybrid retriever achieves a Recall@10 of 0.611, substantially higher than a text-only SBERT retriever (0.579) or a graph-only retrieval (0.260). On the CQA-12k dataset, the hybrid method also obtains the highest Recall@10 in all the question categories, with particularly notable improvements for simple factoid questions and those requiring a logical combination of multiple facts. These gains indicate that our system successfully leverages complementary evidence: the knowledge graph often contributes precise factual triples, while the vector search brings in explanatory or connecting information from text, resulting in better coverage of the answer information.

The main contributions of this work are as follows:

1. **Hybrid Retrieval Pipeline**: We design a complete QA retrieval pipeline that integrates a Neo4j knowledge graph backend with a ChromaDB vector search engine (with the text embeddings), demonstrating an effective way to combine structured and unstructured data for evidence retrieval.
2. **Dynamic Evidence Fusion**: We introduce a novel fusion technique based on Dice–Sørensen similarity to dynamically merge and rank knowledge graph triples and text passages. This approach is original in using lightweight string similarity for hybrid evidence integration, as opposed to training complex rankers or joint representations.
3. **Experimental Evaluation**: We provide an extensive evaluation on two QA benchmarks (WebQSP and CQA-12k), showing that the hybrid approach improves retrieval recall and ranking (Recall@10, MRR, nDCG@10) over single-source base-

[1]The WebQSP dataset used in this paper can be accessed at https://huggingface.co/datasets/rmanluo/RoG-webqsp - last accessed on July 14, 2025

lines. We analyze performance across question types, highlighting where hybrid retrieval is most beneficial.

4. **Practical Insights**: We discuss the computational efficiency of our pipeline, which can run on a consumer-grade GPU with modest memory, and its applicability to real-world scenarios. We also identify opportunities for qualitative error analysis and outline future work, including using retrieved evidence to enable the generation (Retrieval-Augmented Generation - RAG) of answers with smaller language models.

The rest of the paper is organized as follows. Section 2 reviews related work in knowledge graph, text, and hybrid QA retrieval. Section 3 details our hybrid retrieval methodology, including data indexing, query processing, and dynamic fusion. Section 4 presents the experimental setup and quantitative results on WebQSP and CQA-12k. Section 5 provides an analysis of the results, discusses the impact of the fusion strategy, justification of the evaluation metrics, and examples illustrating the system's behavior. Section 6 concludes the paper and suggests future extensions of this work, such as integrating the retriever with generative QA models.

2. Related Work

Early question-answering systems were often specialized for either knowledge base queries or text retrieval.

Knowledge Base QA (KBQA): Techniques in KBQA map natural language questions to structured queries (SPARQL, Cypher, etc.) to fetch answers from a knowledge graph. Notable benchmarks like WebQuestions and WebQuestionsSP use Freebase, and methods have included semantic parsing and relation mapping [Berant et al. 2013] [Yih et al. 2016]. These systems can precisely retrieve factual answers if the query matches the KB content, but they fail when the required fact is not in the KB or the question is not easily translated to a graph query.

Text-based Open-Domain QA: Another line of work uses an information retrieval (IR) component to fetch text passages (e.g., from Wikipedia) and then applies a reading comprehension model to extract answers [Chen et al. 2017, Karpukhin et al. 2020]. Traditional IR uses sparse keyword matching (TF-IDF or BM25), while modern approaches use dense neural retrievers like DPR [Karpukhin et al. 2020] or Sentence-BERT [Reimers and Gurevych 2019] to encode questions and passages into vector embeddings for similarity search. These have achieved strong results in datasets like Natural Questions and TriviaQA. However, pure text retrievers may return irrelevant information if the question requires specific relations or if it can be answered more directly via a knowledge graph lookup. Moreover, text alone might require aggregating information from multiple documents for multi-hop questions [Lewis et al. 2020].

Hybrid QA Systems: To overcome the limitations of using a single knowledge source, recent research has proposed hybrid QA approaches that combine KB and text. One direction is early fusion, where structured and unstructured data are merged into a unified representation. For example, GRAFT-Net [Sun et al. 2018] builds a question-specific graph that includes both retrieved text passages and KB entities/relations and applies graph neural networks to find answers; this achieved significantly better results than using either source alone. Similarly, PullNet [Sun et al. 2019] iteratively expands a

subgraph by "pulling" information from the KB and text corpus, allowing multi-hop reasoning; it showed dramatic improvements, especially when the KB was incomplete. Another approach, Unik-QA [Oguz et al. 2020], converts knowledge base triples into textual form and indexes them together with corpus text in a single dense vector index so that the retriever can return both types of content uniformly.

Other hybrid methods perform late fusion or re-ranking of separate retrieval results. [Zhou et al. 2020] proposed a knowledge-aided open-domain QA (KAQA) framework that first retrieves documents with a text search, then uses external knowledge graphs to re-rank those documents based on inter-document relations. Similarly, KG-FiD [Yu et al. 2021] re-ranks passages by incorporating relationships from a knowledge graph before feeding them to a Fusion-in-Decoder reader.

These methods demonstrated improved retrieval and answer accuracy by ensuring that the retrieved contexts are not only individually relevant but also collectively consistent with a knowledge graph. However, many of the above systems require complex architectures (e.g., training graph neural networks or dual-encoder models) and significant computational resources for training or inference. In contrast, our work focuses on a simpler hybrid retrieval pipeline that can be implemented with off-the-shelf tools (Neo4j, embedding search) and a lightweight fusion algorithm.

3. Formalization

We formalize the hybrid retrieval framework as a combination of two parallel retrieval branches: a **vector retrieval branch** operating in a dense embedding space, and a **graph retrieval branch** operating over a structured knowledge graph (KG). Each branch returns a set of candidate results with an associated relevance score to the query. We then define a fusion mechanism to merge and rank these candidates for question answering.

3.1. Vector Retrieval (Dense Embeddings)

Indexing. Before doing the retrieval, all the textual representations of candidates (e.g. KG entity descriptions or associated text) are indexed utilizing a pre-trained LM (such as Sentence-BERT). To elaborate, let $x_n \in \mathcal{D}$ a textual representation of a candidate, we apply the LM to x_n to map it to a dense embedding vector:

$$z_n = \mathbf{LM}(x) \in \mathbb{R}^d, \tag{1}$$

where d is the dimension of the embedding.

Retrieval. The vector retrieval employs the same encoding used in the indexing step to encode the question ($z_q = \mathbf{LM}(q) \in \mathbb{R}^d$, where q is the question and z_q its vector representation - embedding). Thus, it uses a k-nearest neighbors retrieval approach to score the most similar candidates. To elaborate, let $s_{\text{vec}}(q, x) = \cos(z_q, z_x)$ be the score of a candidate x for question q:

$$V_k = \{(x_n, s_{\text{vec}}(q, x_n)) \mid x_n \in \underset{x' \in D}{\text{argtopk}} \cos(z_q, z_{x'})\} \tag{2}$$

where V_k is the candidate set of the top k entries from the dataset D paired with their respective cosine similarity scores.

3.2. Graph Retrieval (Knowledge Graph Triples)

Let the knowledge graph $\mathcal{G}$ be a set of triples (h, r, t), where h (head) and t (tail) are entities and r is a relation type. Each triple can be viewed as a structured fact "`(h)-[r]->(t)`". Each element has an associated textual label or name (as part of the textual dataset used for the vector retrieval - Section 3.1 - denoted as x_h, x_r and x_t). The graph retrieval branch gets the triples where the head entity is presented by the question. A triple $T = (h, r, t)$ is scored by measuring the lexical overlap between the query q and the triple's components. A tokenization function is defined as $\mathcal{T}(x)$ that yields the set of tokens (e.g. words or character bigrams) in a string x. Let $\mathcal{T}(q)$ be the tokens of the query and $\mathcal{T}(x_h), \mathcal{T}(x_r), \mathcal{T}(x_t)$ be the tokens of the textual representation of the head (h), edge (r) and tail (t) of a given triple T. The Dice–Sørensen coefficient is used to compute a similarity score between these token sets:

$$s_{\text{graph}}(q, T) = \frac{\text{Dice}\left(\mathcal{T}(q), \mathcal{T}(x_h)\right) + \text{Dice}\left(\mathcal{T}(q), \mathcal{T}(x_r)\right) + \text{Dice}\left(\mathcal{T}(q), \mathcal{T}(x_t)\right)}{3} \quad (3)$$

The selection of the Dice-Sørensen coefficient as the lexical similarity metric (s_{graph}) is justified both empirically and theoretically. In preliminary experiments, the Dice-Sørensen coefficient consistently outperformed other string similarity functions available in Neo4j, such as the edit-distance metrics Levenshtein and Jaro-Winkler. This empirical finding is well-aligned with the nature of the task; while edit-distance metrics are effective for single terms or misspellings, they are less adept at measuring relevance between multi-word phrases. In contrast, the Dice-Sørensen coefficient, as a set-based metric operating on tokens, directly quantifies shared vocabulary, making it robust to word order and highly effective at capturing the lexical overlap that signals a potential answer. Its proven performance and theoretical suitability for this context thus validate its choice for the graph retrieval branch.

The graph retrieval branch returns a ranked list G_k of tail entities of the top k triples from $\mathcal{G}$ with the highest $s_{\text{graph}}(q, T)$.

Hybrid Fusion and Ranking As illustrated in Figure 1, the hybrid retrieval framework comprises two sequential branches—a **vector retrieval branch** and a **graph retrieval branch**—whose outputs are merged and sorted to produce the final answer candidates. In hybrid mode, the graph branch is explicitly *seeded* by the entities retrieved from the vector branch:

$$E_{\text{seed}} = \left\{ e \mid \left(e,\ s_{\text{vec}}(q, e)\right) \in V_{\text{k}} \right\},$$

and restrict graph search to triples emanating from any $e \in E_{\text{seed}}$. Having obtained the two result sets V_k and G_k, a unified pool is formed as $R_{\text{fusion}} = V_k \cup G_k$.

Each candidate x in R_{fusion} carries exactly one similarity score—either $s_{\text{vec}}(q, x)$ if retrieved by the vector branch, or $s_{\text{graph}}(q, x)$ if retrieved by the graph branch.

For the final ranking, R_{fusion} is sorted in descending order of a fused score $s_{\text{final}}(q, x)$. To resolve the unlikely event of a perfect tie between a graph-based result and a vector-based result, we introduce a minute bias $\epsilon > 0$ (e.g. $\epsilon = 10^{-6}$) in favor of the

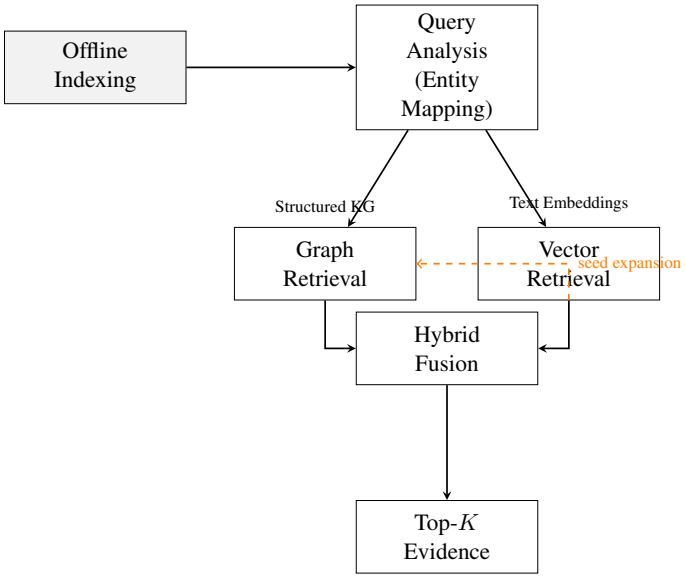

Figure 1. Hybrid retrieval pipeline: (1) offline indexing of text and graph, (2) query analysis with entity mapping, (3) dense and graph retrieval with seed expansion from the dense branch into the graph, (4) fusion of evidence, (5) top-K selection.

graph branch. Concretely, we define:

$$s_{\text{final}}(q,x) = \begin{cases} s_{\text{graph}}(q,x) \;+\; \epsilon\,, & x \in G_k, \\ s_{\text{vec}}(q,x)\,, & x \in V_k. \end{cases} \tag{4}$$

Sorting all candidates by $s_{\text{final}}(q,x)$ yields the merged ranked list of top-K items. In summary, the hybrid retriever returns a unified ranking of entities or triples drawn from both the embedding space and the KG, scored by their relevance to the query. This formulation is model-agnostic: any dense embedding model and any set-similarity function can substitute for the cosine and lexical measures, respectively. The key idea is that s_{vec} captures semantic similarity beyond exact wording, while s_{graph} ensures structural and lexical alignment with the KG, and the fusion s_{final} balances the two.

4. Proposed Hybrid Approach

This section describes our concrete implementation of the hybrid retrieval method for question answering over knowledge graphs. First, the textual corpus is ingested into ChromaDB by segmenting documents into passages, computing SBERT embeddings for each passage, and building a vector-index for efficient nearest-neighbour search. In parallel, the knowledge graph is imported into Neo4j: nodes and relationships are loaded as property graphs, relevant predicates are indexed, and APOC procedures are enabled to support subgraph extraction and similarity functions.

After the preparation step described above, the proposed system follows the abstract framework (described in Section 3) with specific model and tool choices. It customizes graph query formulation and retrieval fusion for two QA benchmarks (WebQSP and CQA-12k - detailed in Section 5.1) by adapting the Cypher templates (as shown in

Figure 2) and passage selection criteria to each dataset's schema and reasoning requirements.

Vector Branch Implementation. We use Sentence-BERT embeddings to power the dense retrieval branch. In particular, we adopt the `all-MiniLM-L6-v2` model – a 6-layer MiniLM-based SBERT model that maps text to a 384-dimensional semantic vector. All textual content (e.g. entity names or descriptions from the knowledge graph) is encoded into vectors offline. We index these vectors using ChromaDB, an open-source vector database that supports fast k-nearest-neighbor queries in the embedding space. At query time, the input question q is encoded into $\mathbf{e}(q)$ using the same SBERT model. We then perform a similarity search in ChromaDB: the database returns the top-K_1 stored items whose embeddings are closest to $\mathbf{e}(q)$ under cosine similarity. This yields the set V_k as defined in Section 3.1.

In this setting, each element of the set V_k corresponds to a KG entity based on its related text snippet. For example, if the question is "Who directed Inception?", the vector retriever might return the entity *Inception* (the film) as a top candidate because the query mentions "Inception," or even the entity *Christopher Nolan* if there is any text passage that mentions he "directed Inception." These retrieved entities (identified by some ID or name) will serve as seeds for the graph search. We note that vector retrieval handles synonyms and semantic paraphrasing – e.g. a question phrased differently but referring to the same concept can still retrieve the relevant entity – which complements the exact-match nature of graph queries.

Graph Branch Implementation. For the structured retrieval branch, we use a Neo4j graph database to store the knowledge graph and execute retrieval queries. We leverage Neo4j's Cypher query language with the APOC procedure library, which provides multiple text similarity functions. We decide to use the Dice–Sørensen string similarity (`apoc.text.sorensenDiceSimilarity`) since it performed around 1.5-2.5% better than the other functions we've tried (`levenshteinSimilarity` and `jaroWinklerDistance`) in our tests. Rather than scanning the entire KG for matches, we perform a focused seed expansion: we restrict graph retrieval to entities that were identified by the vector branch. Specifically, let $E_{\text{seed}} = e_1, e_2, \ldots, e_{K_1}$ be the set of top entities (by ID) returned by the vector search. We then query the graph for any triples from these seed entities and rank them by lexical similarity to the question. In Cypher, this looks like the code presented in Figure 2.

As per shown in Figure 2, `r.description` and `t.label` denote the relation's textual description and the object entity's name, respectively; `seeds` is the array of entity IDs, provided by the vector retrieval; and q is the query–string parameter. The result is the set of triples emanating from the seed entities, ordered by their similarity score to the question. After this graph retrieval step, the pipeline extracts a set of candidate entities (the tail of each triple) G_k.

Fusion of Results. Once both branches have returned their candidates, the pipeline performs the fusion as formalized in Section 3. In implementation, it simply takes the union

```
UNWIND $seeds AS sId
MATCH (s {id:sId})-[r]-(t)
WITH s, r, t,
  apoc.text.sorensenDiceSimilarity(toLower($q),
     toLower(s.label)) AS simS,
  apoc.text.sorensenDiceSimilarity(toLower($q),
     toLower(r.description)) AS simR,
  apoc.text.sorensenDiceSimilarity(toLower($q),
     toLower(t.label)) AS simT
WITH s, r, t, (simS + simR + simT) / 3.0 AS score
ORDER BY score DESC
LIMIT $limit
RETURN s.name AS subject, type(r) AS predicate, t.name AS
   object, t.name AS entity_id, score
```

Figure 2. Template of a Cypher query to get the similar related entities based on a lexical similarity

of V_k and G_k and sorts all candidates by their score in descending order. Because s_{vec} and s_{graph} are on a comparable scale (both are similarities between 0 and 1), it ranks all results together by score. If an item appears in both lists (for instance, if a particular answer entity was directly retrieved by the vector index and also surfaced via a KG triple), it unifies them and could choose the higher of the two scores as its final score[2].

5. Experimental Results

The experiments were conducted on two datasets to evaluate the effectiveness of the hybrid retrieval pipeline: **WebQuestionsSP (WebQSP)** and **CQA-12k**. Our evaluation focuses on validating the efficacy of a lightweight, training-free, and modular retrieval paradigm. Consequently, we benchmark our method against strong, widely used single-source baselines to rigorously isolate the performance gains attributable to our hybrid fusion strategy.

We acknowledge the existence of powerful, heavyweight hybrid systems like GRAFT-Net and KG-FiD. A direct quantitative comparison was deemed out of scope, as these models represent a different design philosophy. For instance, early-fusion methods like GRAFT-Net build a unified graph from text and KB sources and require training complex Graph Neural Networks for reasoning, which entails significant computational cost. Other approaches like KG-FiD are tightly coupled with specific learned re-rankers or large generative models.

In contrast, our work intentionally prioritizes efficiency, modularity, and transparency. By employing a late-fusion strategy with a simple, unsupervised similarity metric, our pipeline avoids learned parameters and runs on commodity hardware in under 50 milliseconds. The goal is to establish a strong, practical baseline that demonstrates the power of synergizing structured and unstructured data without the overhead of end-to-end training. We compare the following four retrieval modes:

[2]In this system, such overlaps are infrequent, but this step prevents duplicate answer candidates.

1. **Graph-only:** Query the knowledge graph and return up to K facts, without any text retrieval.
2. **Vector-only:** Retrieve the top K passages from the text corpus using SBERT embeddings.
3. **BM25 (Elasticsearch):** Retrieve the top K passages via BM25 ranking function (for comparison purposes).
4. **Hybrid:** Execute the full pipeline, seeding the graph-retrieval branch with entities from SBERT-based text retrieval and then fusing both result sets.

5.1. Datasets

WebQuestionsSP (WebQSP). WebQSP comprises 4,737 natural-language questions each paired with a gold SPARQL query and its answer set drawn exclusively from Freebase. When linearized into triples, the average answer-containing subgraph spans approximately 180–200 triples (approx. 533 tokens), and for roughly 27% of topic entities the immediate 1-hop neighborhood exceeds 1,000 triples. To mitigate noise and improve retrieval performance, it is therefore standard practice to extract a focused 2-hop, topic-centric subgraph prior to query execution [Yih et al. 2016].

CQA-12k (Entity-Answer Slice). The original CSQA corpus provides approx. 12,000 questions for each of seven reasoning categories (aprox. 84,000 total). By retaining only those instances whose gold answers map to Wikidata entities or literal values, a subset of around 65,000 questions is obtained. This filtered set covers six core reasoning types: Simple Question (Direct), Logical Reasoning, Quantitative Reasoning (All), Quantitative Reasoning (Count), Comparative Reasoning (All), and Comparative Reasoning (Count)—each represented by approx. 12,000 examples and designed to stress retrieval-centered evaluation over a broad, sparse slice of Wikidata.

5.2. Metrics

We evaluate retrieval quality using three standard information retrieval metrics: Recall@10, Mean Reciprocal Rank (MRR), and nDCG@10 [Roberts 2024]. Our primary focus is Recall@10, as it is pivotal for retrieval-augmented QA: if the evidence is not in the top-K results, the reader has no chance to generate the correct answer. MRR complements this by rewarding higher placement of the first correct item, while nDCG@10 is important for ranking multiple relevant pieces of evidence, a scenario common in multi-hop questions.

Recall@10 (R@10) The fraction of questions for which at least one relevant evidence item is present in the top 10 results.

Mean Reciprocal Rank (MRR) The average of the reciprocal rank ($1/\text{rank}$) of the first correct result, rewarding higher placement of the answer.

nDCG@10 A graded metric that accounts for the position of multiple relevant items, assigning more weight to higher-ranked evidence.

Table 1. Retrieval effectiveness on WebQSP.

Metric	BM25			Vector			Graph			Hybrid		
	R@10	MRR	nDCG	R@10	MRR	nDCG	R@10	MRR	nDCG	R@10	MRR	nDCG
Total	0.336	0.136	0.136	0.580	0.250	0.279	0.260	0.247	0.313	**0.611**	**0.281**	**0.352**

5.3. Results

As mentioned, for SBERT, we used the pre-trained model `all-MiniLM-L6-v2` for efficiency. The ChromaDB index uses approximate nearest neighbor search (HNSW [Malkov and Yashunin 2018]) to handle the semantic vector lookups quickly. The Neo4j graph was run on the same machine; typical query times for one-hop neighbors or simple two-hop traversals were on the order of a few milliseconds. The entire retrieval pipeline (graph query + vector search + fusion) takes on average under 50 milliseconds per question (200 retrievals per second) on a machine with a 20-core CPU and an NVIDIA RTX 4070 12GB GPU (used for embedding the question on the fly). This indicates the system can potentially scale to real-time QA scenarios, making it a practical alternative to heavyweight neural re-rankers or GNN-based methods. We set $K = 10$ for output evidences. These parameters were chosen empirically on a small validation set (for CQA-12k we used the dev set to ensure these settings made sense).

5.4. Results on WebQSP

Table 1 summarizes the performance on the WebQSP full set. The hybrid retrieval achieves the highest Recall@10, successfully retrieving the answer-containing evidence for 61.1% of the questions in the top 10, compared to 57.9% for the vector-only retriever and only 26.0% for the graph-only approach. This large gap for graph-only is expected: many WebQSP questions require a chain of two relations, and our graph-only baseline that only returns one-hop facts from the seeds entities often doesn't directly include the final answer. The vector approach does better by retrieving a relevant passage (for example, a sentence that combines those two hops of information). The hybrid method, by combining both, not only covers cases where either method would work individually, but in some cases finds the answer via complementary evidence.

The MRR metric shows that hybrid also slightly improves the rank of the first relevant hit (MRR 0.281 vs 0.250 for vector). This means our fusion ranking helped surface relevant info slightly higher on average. The nDCG@10 was similarly highest for hybrid, indicating that when multiple relevant pieces existed (which can happen if, say, both a triple and a passage contain the answer), the hybrid can retrieve both, yielding a higher cumulative gain.

In analyzing WebQSP results, we found that the questions where hybrid had an edge often involved a need for both sources. For example, one question asks: "What river does the bridge named London Bridge cross?" The knowledge graph (Freebase) knows a relation `bridge -> crosses -> river`, but if our entity linker gets "London Bridge" as the topic, the graph might directly give the answer "River Thames". In this case, graph-only would succeed. For a question like "Who is the wife of the actor who portrayed James Bond in Skyfall?", answering requires two hops: find the actor who portrayed James Bond in Skyfall (which is Daniel Craig, presumably in the graph via a

Table 2. Retrieval effectiveness on CQA-12k. The hybrid system outperforms all single–source baselines.

Category	BM25 (Elasticsearch)			Vector (SBERT)			Graph			Hybrid		
	R@10	MRR	nDCG	R@10	MRR	nDCG	R@10	MRR	nDCG	R@10	MRR	nDCG
Comparative (All)	0.276	0.093	0.051	0.409	0.240	0.110	0.057	0.021	0.010	**0.424**	**0.188**	**0.094**
Compar. Count	0.297	0.099	0.053	0.400	0.224	0.101	0.055	0.019	0.009	**0.416**	**0.180**	**0.087**
Quantitative (All)	0.070	0.022	0.013	0.133	0.056	0.043	0.008	0.001	0.001	**0.135**	**0.054**	**0.041**
Simple (Direct)	0.055	0.010	0.019	0.065	0.036	0.041	0.766	0.667	0.693	**0.791**	**0.676**	**0.718**
Quant. Count	0.100	0.032	0.026	0.160	0.075	0.060	0.225	0.151	0.145	**0.353**	**0.194**	**0.183**
Logical Reasoning	0.098	0.029	0.030	0.126	0.063	0.050	0.712	0.533	0.499	**0.765**	**0.550**	**0.523**
Total	0.144	0.046	0.031	0.207	0.111	0.066	0.313	0.242	0.236	**0.483**	**0.313**	**0.283**

film-cast relation), then find his wife. Our graph retrieval would return Daniel Craig's relations (including spouse = Rachel Weisz), but it might not retrieve Daniel Craig in the first place because the topic entity in the question was not directly given (the question didn't name Daniel Craig, it described him). The vector retriever, however, might pull a passage about Skyfall or James Bond that mentions Daniel Craig. In our hybrid approach, the text passage mentioning Daniel Craig could be retrieved, and separately the graph part might retrieve Daniel Craig's spouse given the entity. By fusing, we ended up with Rachel Weisz as an answer evidence. This is an example of how the two sources complement each other: text helped identify the hidden entity needed, and graph provided the final fact.

5.5. Results on CQA-12k

Overall Performance. Table 2 shows that the *hybrid* retriever outperforms every single-source baseline on **all** CQA-12k categories, yielding the best Recall@10, MRR, and nDCG_{10}. The fusion of dense semantics with KG structure therefore provides a uniformly stronger evidence pool than either branch alone.

Why Hybrid Helps. For single-fact queries, (*Simple*), graph lookup already covers most answers, but hybrid seeding recovers the few semantically mismatched triples and pushes recall to 0.791. In contrast, logical and multi-hop questions (*Logical*) benefit from combining partial cues: dense retrieval surfaces entities that satisfy one constraint, the KG supplies the complementary relation, and fusion raises recall from 0.712 (graph) and 0.126 (vector) to 0.765. Comparative questions reveal a complementary pattern: dense retrieval captures paraphrased superlatives ($R@10 = 0.409$), whereas the KG branch contributes explicit numeric attributes ($R@10 = 0.057$); combining the two raises recall to 0.424.

Remaining Challenges. Hybrid gains are smaller in purely quantitative categories, where correct answers require arithmetic or aggregation not handled by retrieval alone. Nevertheless, the hybrid still edges out the best single branch (e.g. 0.135 vs. 0.133 recall in *Quantitative*) by combining numeric statements in text with value-bearing triples in the graph. These results confirm that dense and symbolic evidence are complementary: dense embeddings offer semantic recall, the KG offers structural precision, and their union consistently yields the highest ranked lists across diverse reasoning types.

6. Conclusion and Future Work

We introduced a training-free, *hybrid* evidence retriever that unites a Neo4j knowledge graph with a ChromaDB vector index via a lightweight overlap-based fusion. Experiments on WebQSP and CQA-12k show consistent gains over graph-only, text-only, and BM25 baselines, with the hybrid achieving the best RECALL@10, MRR, and nDCG_{10}. The improvement stems from two complementary effects: (i) dense embeddings recover semantically phrased passages that lexical KG labels miss, and (ii) once those entities are used as seeds, the KG supplies precise triples. The fusion, implemented as a Dice–Sørensen overlap between the query and each candidate's labels, dynamically balances both evidence sources by design: if the query has high lexical overlap with graph relations, structured results are prioritized, otherwise dense passages prevail. This occurs without any learned re-ranker, keeping the pipeline transparent and fast.

Practical reach. Because the method requires only off-the-shelf SBERT encoders and a string-similarity operator in Cypher, it is readily portable to enterprise KGs or domain-specific text collections. The clear scoring logic aids debugging and supports trust in deployed QA systems.

Next steps. (1) *End-to-end QA*: couple the retriever with a lightweight generator (e.g. Qwen 7B or LLaMA 3 8B) and measure exact-match/F1 gains on Retrieval-Augmented Generation (RAG) tasks, such as the ones described by [Xavier and da Silva Soares 2024]. (2) *Learned fusion*: train a small classifier to weight graph vs. text scores per question type, or to expand the synonym space beyond strict token overlap. (3) *Query-aware KG search*: replace one-hop expansion with pattern-guided multi-hop Cypher to cut graph noise. (4) *Reasoning operators*: exploit the KG for on-the-fly aggregation (counts, comparisons) when answers are not explicitly stored.

Although the proposed extensions may yield further improvements in retrieval effectiveness, they inevitably incur additional computational overhead. Consequently, it is essential to strike an appropriate trade-off between accuracy and runtime performance to ensure the solution remains viable for deployment on enterprise-scale knowledge graphs.

Overall, the study underscores that structured and unstructured knowledge are *synergistic*; a simple, interpretable fusion can harvest that synergy without heavy learning infrastructure, paving the way for cost-efficient yet high-coverage QA systems.

References

Berant, J., Chou, A., Frostig, R., and Liang, P. (2013). Semantic parsing on Freebase from question-answer pairs. In Yarowsky, D., Baldwin, T., Korhonen, A., Livescu, K., and Bethard, S., editors, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Chen, D., Fisch, A., Weston, J., and Bordes, A. (2017). Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Ju, M., Yu, W., Zhao, T., Zhang, C., and Ye, Y. (2022). Grape: Knowledge graph enhanced passage reader for open-domain question answering. *arXiv preprint arXiv:2210.02933*.

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. (2020). Dense passage retrieval for open-domain question answering. In *Proceed-*

ings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 6769–6781.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Petrov, S. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Malkov, Y. A. and Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836.

Oguz, B., Chen, X., Karpukhin, V., Peshterliev, S., Okhonko, D., Schlichtkrull, M., Gupta, S., Mehdad, Y., and Yih, S. (2020). Unik-qa: Unified representations of structured and unstructured knowledge for open-domain question answering. *arXiv preprint arXiv:2012.14610*.

Reimers, N. and Gurevych, I. (2019). 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)*, pages 3982–3992.

Roberts, K. (2024). *Information Retrieval*, pages 195–230. Springer International Publishing, Cham.

Saha, A., Pahuja, V., Khapra, M. M., Sankaranarayanan, K., and Chandar, S. (2018). Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph.

Sarmah, B., Mehta, D., Hall, B., Rao, R., Patel, S., and Pasquali, S. (2024). Hybridrag: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction. In *Proceedings of the 5th ACM International Conference on AI in Finance*, pages 608–616.

Sun, H., Bedrax-Weiss, T., and Cohen, W. (2019). PullNet: Open domain question answering with iterative retrieval on knowledge bases and text. In Inui, K., Jiang, J., Ng, V., and Wan, X., editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2380–2390, Hong Kong, China. Association for Computational Linguistics.

Sun, H., Dhingra, B., Zaheer, M., Mazaitis, K., Salakhutdinov, R., and Cohen, W. (2018). Open domain question answering using early fusion of knowledge bases and text. In Riloff, E., Chiang, D., Hockenmaier, J., and Tsujii, J., editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242, Brussels, Belgium. Association for Computational Linguistics.

Xavier, O. C. and da Silva Soares, A. (2024). Geração com recuperação aumentada (rag) em grafos de conhecimento. In da Silva Monteiro Filho, J. M., Razente, H., and dos Santos Mello, R., editors, *Tópicos em Gerenciamento de Dados e Informações: Minicursos do SBBD 2024*. Sociedade Brasileira de Computação, São Paulo, Brazil.

Yih, W.-t., Richardson, M., Meek, C., Chang, M.-W., and Suh, J. (2016). The value of semantic parse labeling for knowledge base question answering. In *Proceedings of*

the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 201–206.

Yu, D., Zhu, C., Fang, Y., Yu, W., Wang, S., Xu, Y., Ren, X., Yang, Y., and Zeng, M. (2021). Kg-fid: Infusing knowledge graph in fusion-in-decoder for open-domain question answering. *arXiv preprint arXiv:2110.04330*.

Zhou, M., Shi, Z., Huang, M., and Zhu, X. (2020). Knowledge-aided open-domain question answering. *arXiv preprint arXiv:2006.05244*.