OmniRetrieval: Unified Retrieval across Heterogeneous Knowledge Sources

Anonymous ACL submission

Abstract

Real-world information needs require access to structurally diverse knowledge sources, from unstructured text and relational tables to knowledge graphs and property graphs. Existing retrievers, however, operate over one source at a time under a fixed query language, leaving the broader landscape of available knowledge fragmented behind incompatible interfaces. A natural attempt at unification would collapse these sources into a shared space, but this erases the structural affordances (such as schemas, ontologies, compositional operators) that give each source its expressive power. Effective retrieval over diverse knowledge, therefore, requires not homogenization but an overarching layer that meets each source on its own terms. To achieve this, we present OmniRetrieval, a framework that takes any natural-language query, identifies appropriate knowledge sources, and dispatches source-native queries to their native execution engines. Across an extensive benchmark spanning 13 datasets and 309 distinct knowledge bases over text, relational, and graph-structured sources, OmniRetrieval exceeds single-source baselines, demonstrating that it can serve as a general-purpose interface to the heterogeneous sources while preserving the structural distinctions that make each source valuable.

1 Introduction

The knowledge that answers a real-world question rarely lives in a single place, or in a single shape. A clinical question may be answered by a passage in a biomedical article (Thakur et al., 2021); an enterprise question may require a join across normalized relational tables (Yu et al., 2018; Li et al., 2023); a factoid question about people, places, or events may resolve to a few triples in an encyclopedic knowledge graph (Bollacker et al., 2008; Vrandecic and Krötzsch, 2014); and a question about a supply chain or an academic collaboration network may turn on a multi-hop traversal of a labeled property graph (Ozsoy et al., 2025). In each case, the answer is retrievable in principle, but only if one knows which corpus to consult, which query language to write, and which execution engine to dispatch it to. The retrieval problem, then, is not merely to find relevant content within a source, but to navigate the structural heterogeneity across sources.

Existing retrieval approaches, however, are typically designed for one source at a time. Specifically, document retrievers operate over an unstructured corpus and rank passages by similarity to a free-form query (Robertson et al., 1994; Karpukhin et al., 2020); text-to-SQL systems target a single relational database and emit a single SQL dialect (Yu et al., 2018; Li et al., 2023); SPARQL or Cypher generators are likewise tied to a single graph backend and query language, with SPARQL for RDF stores and Cypher for labeled property graphs (Brei et al., 2025; Ozsoy et al., 2025). As a consequence, even when a recent Large Language Model (LLM) can reason across evidence drawn from many kinds of sources (Anthropic, 2024; Gemini Team, 2025; OpenAI, 2026), the retrieval layer that feeds it cannot reach into them all, leaving the broader knowledge landscape out of reach.

A natural response is to collapse the silos themselves, by projecting every knowledge source into a shared representation, typically a single dense embedding space or a shared linearized text format (Oguz et al., 2022; Ma et al., 2022; Baek et al., 2023). However, this restores a uniform interface at a cost: the structural affordances that distinguish each source are flattened away, and what remains is a lossy projection with two consequences. First, the unified embeddings cluster by source type rather than by semantic content, a modality gap that biases retrieval toward sources resembling the query in form rather than those that actually answer it (Yeo et al., 2025). Second, it supports only the similarity matching, and the native query operations of each source are lost. Inspired by these, the

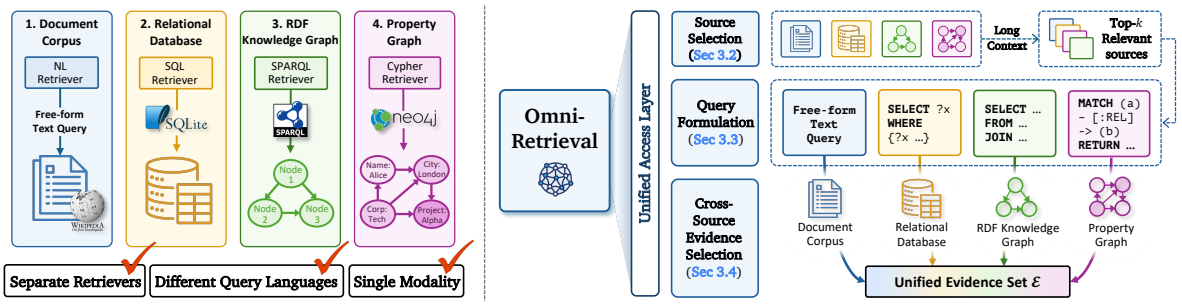

Figure 1: Different knowledge sources offer distinct structural affordances and query languages (left); OmniRetrieval meets each on its own terms via source selection, native query formulation, and cross-source condensation (right).

right move, we argue, is the opposite of homogenization: keep each source on its own terms, and instead build a unifying *access layer* above them.

In this work, we instantiate this view in **OmniRetrieval**, a framework that engages the knowledge sources relevant to a query, each through its own query language. Specifically, given a query, OmniRetrieval identifies which of the available sources are relevant, and for each formulates an executable query in the corresponding native language, conditioning on whatever structural context the source exposes. These queries are then executed by the sources themselves, and when more than one is engaged, the outputs are consolidated to retain those relevant to the question. Notably, as every source is reached on its own terms rather than through a shared representation, adding a new base is a matter of registration alone, with no shared encoder to retrain and no embedding space to redraw.

We then evaluate OmniRetrieval on an extensive benchmark that spans 309 distinct knowledge bases across four backend types (such as unstructured corpora, relational databases, RDF knowledge graphs, and labeled property graphs), drawn from 13 publicly available datasets (Bordes et al., 2015; Yu et al., 2018; Dubey et al., 2019; Thakur et al., 2021; Li et al., 2023; Usbeck et al., 2024; Ozsoy et al., 2025). Across this suite, OmniRetrieval consistently exceeds single-source baselines constrained to operate within one modality, while also identifying the correct knowledge bases for each query with high accuracy and producing structurally valid queries in each native language. These results affirm that unified retrieval across distinct knowledge sources can be achieved by coordinating access through the native interfaces of each source.

2 Related Work

Retrieval over Heterogeneous Sources Classical retrieval has long been organized around a single corpus and representation, from lexical retrievers that rank passages by term overlap (Robertson et al., 1994; Jones, 2004) to dense retrievers that project queries and documents into an embedding space (Karpukhin et al., 2020; Xiong et al., 2021), with multi-modal extensions adding specific encoders for images or video (Radford et al., 2021; Jeong et al., 2025b). To lift this single-source restriction, one line of work collapses heterogeneous sources, such as text passages, knowledge graph facts, and tabular records, into a shared representation so that a single retriever can rank items across them (Oguz et al., 2022; Ma et al., 2022; Baek et al., 2023). Other efforts cover either structured or unstructured sources but not both: query-type routers are confined to unstructured corpora and rely on embedding similarity within each (Yeo et al., 2025), while hand-crafted, per-type interface functions are confined to structured sources and further presuppose a single target source fixed at input rather than chosen per question, with native query generation only for relational databases (Jiang et al., 2023). Yet none of these approaches generate queries uniformly in the native language of each backend: structure is flattened into a common representation, surfaced only through embedding similarity, or fragmented across per-type interface functions with predefined extraction primitives. In contrast, OmniRetrieval spans both structured and unstructured backends with a native query for each, preserving the symbolic operators each source exposes.

Native-Language Query Generation A substantial body of work studies how to translate a natural-language query into the native query language of a single backend. Text-to-SQL has matured around schema-grounded generation over relational databases, with widely used benchmarks (Yu et al., 2018; Li et al., 2023) and capable LLM-based generators that perform schema linking and SQL synthesis (Rajkumar et al., 2022; Gao et al., 2024).

Parallel lines of work address text-to-SPARQL for RDF stores (Drozdov et al., 2023; Schneider et al., 2024; Brei et al., 2025) and text-to-Cypher for labeled property graphs (Guo et al., 2023; Ozsoy et al., 2025), each respecting the syntax and semantics of the target language. OmniRetrieval inherits this per-backend capability and lifts it into a multi-backend setting, where any existing generator can be plugged in as the query-formulation module, while introducing challenges intrinsic to this regime: joint source selection, query formulation across multiple native languages within one model, and cross-form evidence consolidation.

LLMs as Tool-Use Agents A broader paradigm casts LLMs as agents that interact with external tools via API calls to extend their capabilities beyond parametric memory (Yao et al., 2023; Schick et al., 2023; Qin et al., 2024; Patil et al., 2024). OmniRetrieval shares this dispatch pattern but is, to our knowledge, the first to unify retrieval across structured and unstructured backends through their native query languages, and differs from generic tool use. First, each invocation is program synthesis over a backend schema spanning hundreds of tables or thousands of relations, well beyond the fixed function signatures typical of tool use. Also, results are heterogeneous (passages, table rows, KG triples, graph paths) and require cross-form consolidation, unlike independent tool outputs.

3 Method

3.1 Problem Formulation

Let q be a natural-language question from a user and $\mathcal{B} = \{b_1, \ldots, b_N\}$ be a pool of independently maintained knowledge sources. Each of these sources $b \in \mathcal{B}$ has its own native query language (such as SQL for a relational database, SPARQL for an RDF graph, Cypher for a labeled property graph, or free-form text for an unstructured corpus), its own execution engine $\texttt{Exec}(b, \hat{q})$ that accepts a native query $\hat{q}$ (written in that language) and returns a set of results, and an exposed structural context c_b (such as a relational schema, an ontology, or a corpus descriptor) that any external caller can read in order to formulate an executable query against b. However, knowledge sources may differ arbitrarily in what they store and how they store it, where one may hold unstructured text, another normalized tables, and a third a labeled graph, and they return their results in correspondingly different forms.

The retrieval task is then to find and provide, for the question q, a set of evidence drawn from one or more sources in $\mathcal{B}$ that is relevant to q. Notably, a retrieval framework addressing this task should operationalize the selection of a subset $\mathcal{S} \subseteq \mathcal{B}$ of sources to engage, the formulation of an executable query $\hat{q}_b$ in the native language of each $b \in \mathcal{S}$, and the consolidation of the executor outputs $\{\texttt{Exec}(b, \hat{q}_b)\}_{b \in \mathcal{S}}$ into a single evidence set relevant to q. This formulation has clear strengths. In particular, since each source is engaged through its own native language, the structural operators it exposes (such as joins, traversals, property paths) are preserved rather than approximated by similarity in a shared space. Also, keeping each source on its own terms makes adding a new source a matter of registration rather than infrastructure rebuilding, and lets the framework draw on any of its registered sources for a single question (whether the answer comes back as a passage, a tuple, a triple, or a path) without committing to one backend up front.

3.2 Source Selection

We now turn to the first operation in our OmniRetrieval framework, which identifies a subset $\mathcal{S} \subseteq \mathcal{B}$ of sources to engage for a question q.

Challenges in Source Selection The set of registered sources $\mathcal{B}$ can be large and is open-ended, since new sources are added by registration alone, and the structural contexts $\{c_b\}_{b \in \mathcal{B}}$ that distinguish one source from another are heterogeneous in form (a schema lists tables and columns, an ontology declares classes and predicates, and a corpus descriptor characterizes the topics and style of its documents). One straightforward approach to operationalize selection is to embed each c_b and the query q into a shared vector space and rank sources by similarity, following standard single-corpus retrieval practice. However, this approach is restrictive in our setting, since the descriptors are not uniform in form so a single encoder cannot represent them without lossy projection, and the decision of whether b can answer q often hinges on the actual contents of c_b (such as a table name in a relational schema, or a relationship type in a property graph), which a similarity score alone cannot capture.

Long-Context Source Selection To sidestep this issue, inspired by recent works showing that long-context LLMs can retrieve and reason directly over textual inputs at the scale of entire corpora (Lee et al., 2024; Jeong et al., 2025a), we propose to read the full catalog of source descriptors jointly

with q and identify the sources to engage. However, in contrast to such prior works that leverage this in-context capability to homogeneous corpora, our approach involves heterogeneous knowledge sources whose descriptors each take a different form. Formally, a long-context LLM takes the query q and the structural descriptors $\{c_b\}_{b\in\mathcal{B}}$ of all registered sources (e.g., schemas, ontologies, and corpus summaries; see Appendix A for concrete examples) as input, and returns a ranked subset of $\mathcal{B}$:

$$\mathcal{S} = \texttt{LLM}_{\texttt{select}}(q,\ \{c_b\}_{b\in\mathcal{B}};\ k) \subseteq \mathcal{B}, \qquad (1)$$

where the LLM returns at most k sources ordered by relevance to q. Since c_b here is the same structural context each source already publishes under the formulation, it could be used as-is and grows by simply appending a new descriptor. More interestingly, the operator returns a short list of candidates, which enables accommodating the queries that require multiple sources as well as the queries whose target source is ambiguous, with the final decision then deferred to the evidence selection stage, where it can rest on the retrieved evidence.

3.3 Query Formulation

Given $\mathcal{S}$, we now formulate an executable query $\hat{q}_b$ in the native language of each source $b \in \mathcal{S}$.

Challenges in Query Formulation The knowledge sources in $\mathcal{S}$ each speak their own native query language, each shaped by the structure of the data it queries: SQL expresses joins and set operations over normalized relational tables, SPARQL matches triple patterns over an RDF graph, Cypher traverses paths over the labeled nodes and relationships of a property graph, and free-form text drives similarity-based retrieval over a corpus. Beyond the difference in languages, an executable query for b should also refer to the elements that b actually exposes, such as its specific tables and columns in a relational schema, the predicates declared in an RDF ontology, the relationship types of a property graph, or the topical scope and style of the corpus. The proposed framework should therefore produce, for every source $b \in \mathcal{S}$, a query that is both valid in its native query language and grounded in its structural context c_b, across distinct backend types.

Per-Source Native Query Generation To address this, for each $b \in \mathcal{S}$ the framework translates the question q into a native query $\hat{q}_b$ in the language of b, conditioned on the structural context c_b:

$$\hat{q}_b = \texttt{Generate}_b(q,\ c_b) \quad \text{for each } b \in \mathcal{S}. \qquad (2)$$

Here, we instantiate $\texttt{Generate}_b$ as $\texttt{LLM}(\mathcal{T}_b(q,\ c_b))$, with $\texttt{LLM}$ as a single LLM shared across all sources and $\mathcal{T}_b$ a per-source prompt template that incorporates q, c_b, and an instruction identifying the native query language of b. In particular, for SQL, SPARQL, and Cypher, the LLM emits the executable native query directly; for an unstructured corpus, the retriever accepts free-form text, so q itself can serve as the retriever query, and the LLM could optionally be used to optimize q to improve retrieval. Additionally, the LLM-based realization above is one of several possible instantiations of $\texttt{Generate}_b$, and any method that maps q and c_b to a valid native query for b would fit the framework.

3.4 Cross-Source Evidence Selection

Running each $\hat{q}_b$ through $\texttt{Exec}(b, \cdot)$ yields a collection of executor outputs, from which we now select the final evidence set $\mathcal{E}$ relevant to q.

Role of Selection Our task is retrieval, whose objective is to return what is relevant to the question and filter out what is not; however, the executor outputs do not yet meet that goal. Specifically, running each $\hat{q}_b$ through $\texttt{Exec}(b, \cdot)$ produces an individual retrieval result for each source $b \in \mathcal{S}$, with $\mathcal{S}$ itself possibly spanning multiple sources because the question genuinely requires them or because the source-selection step deferred an ambiguous routing decision here. In addition, these per-source results are heterogeneous in both form (rows from SQL, triples from RDF, paths from property graphs, passages from an unstructured corpus) and size, since each $\texttt{Exec}(b, \hat{q}_b)$ returns results at the granularity of the native query, ranging from many items down to a single value such as an entity or an aggregate, only some of which are typically relevant to q. The role of this step is therefore to pick, from $\{\texttt{Exec}(b, \hat{q}_b)\}_{b\in\mathcal{S}}$, the subset relevant to q, completing the retrieval task by filtering out what is not.

Cross-Source Evidence Selection Formally, the OmniRetrieval framework implements this as an operator $\texttt{Select}$ that takes q and the executor outputs, and then returns the relevant subset $\mathcal{E}$:

$$\mathcal{E} = \texttt{Select}(q,\ \{\texttt{Exec}(b, \hat{q}_b)\}_{b\in\mathcal{S}}). \qquad (3)$$

Here, we instantiate $\texttt{Select}$ as $\texttt{LLM}(\mathcal{T}_{\text{sel}}(\cdot))$, with $\mathcal{T}_{\text{sel}}$ a prompt template that verbalizes each executor output in its native form (rows for SQL, triples for RDF, paths for property graphs, and passages for an unstructured corpus), and asks the model to identify

the outputs relevant to q. It is worth noting that, although native query languages are essential at the query stage to express structural operators (joins, traversals, paths), verbalizing the executor outputs here does not undercut that choice: by this point `Exec` has already done the structural work via those operators, providing results that can be read as text.

4 Experimental Setup

4.1 Datasets and Knowledge Bases

We evaluate OmniRetrieval on a benchmark compiled from 13 datasets that, in combination, span all four native backends, and that together provide a pool of 309 distinct knowledge bases.

Document Search For document retrieval over unstructured corpora, whose task is to identify documents that are most relevant to a natural-language query, we use seven datasets of various domains from the BEIR benchmark (Thakur et al., 2021): NFCorpus (medical) (Boteva et al., 2016), SciFact (scientific claim verification) (Wadden et al., 2020), FiQA (Maia et al., 2018) (financial question answering), MS MARCO (web passages) (Nguyen et al., 2016), FEVER (Wikipedia fact verification) (Thorne et al., 2018), Natural Questions (short-answer question answering) (Kwiatkowski et al., 2019), and HotpotQA (Yang et al., 2018) (multi-hop question answering). Each document collection itself serves as a knowledge base.

Relational Databases For text-to-SQL, the task of translating a natural-language question into a SQL query over a relational database, we use Spider (Yu et al., 2018) and BIRD (Li et al., 2023): Spider brings 206 databases across diverse domains, and BIRD brings a further 80 larger databases from real-world applications, yielding 286 knowledge bases in total, with each provided as a SQLite database against which the SQL is executed.

RDF Knowledge Graphs For text-to-SPARQL, the task of translating a natural-language question into an executable SPARQL query over an RDF knowledge graph (a structured store of subject-predicate-object triples), we use the following three datasets: SimpleQuestions (Bordes et al., 2015) for single-triple factoid questions, QALD-10 (Usbeck et al., 2024) for hand-curated questions covering factoid and aggregation queries, and LC-QuAD 2.0 (Dubey et al., 2019) for large-scale, compositional questions. As the largest publicly-queryable RDF knowledge graph and the standard target of modern SPARQL benchmarks, Wikidata is the single knowledge base in this backend, against whose public SPARQL endpoint[1] the query is executed.

Labeled Property Graphs For text-to-Cypher, the task of translating a natural-language question into a Cypher query over a property graph (whose nodes and edges carry typed labels along with key-value properties dictated by the graph-specific data model), we use Text2Cypher (Ozsoy et al., 2025), which spans 15 graphs from the Neo4j collection, covering various domains (such as movie recommendations, company structures, social networks, and financial investigations), with the generated query executed against the Neo4j Labs endpoint[2].

For each dataset, we sample 300 questions for evaluation. The structural context c_b is a topical descriptor for document collections and a schema for each structured backend; example queries and the verbatim form of each c_b are in Appendix A.

4.2 Methods

We compare OmniRetrieval against baselines that differ in how they engage the pool of sources.

Single-Backend Baselines We first include four baselines that pin the pipeline to a single retrieval paradigm and operate on every query irrespective of the underlying knowledge. Specifically, **Document Search** answers a question through retrieval over an unstructured corpus; **Text-to-SQL** formulates a SQL query against a relational database; **Text-to-SPARQL** formulates a SPARQL query against an RDF knowledge graph; and **Text-to-Cypher** formulates a Cypher query against a labeled property graph. For a fair comparison with our framework, the knowledge base within each paradigm is selected per query in the same manner.

KB Routing We further include a baseline that, unlike the four above, lets the model choose any backend per query: it reads the catalog of source descriptors jointly with the question and routes to a single knowledge base, after which query formulation and execution proceed on that source.

OmniRetrieval This is the proposed framework, which engages multiple candidate sources: given a question, it reads the source catalog and returns a short list of candidates (e.g., 3), formulates a native query for each, executes them, and consolidates the results through cross-source evidence selection.

[1] `https://query.wikidata.org/sparql`
[2] `neo4j+s://demo.neo4jlabs.com:7687`

Method	GPT-5.4	Gemini-3.1 (Pro)	Sonnet-4.6	Qwen-3.5 (27B)	Gemma-4 (31B)	Average
Source Selection Accuracy						
Document Search	21.49	22.61	21.90	21.49	21.42	21.78
Text-to-SQL	14.75	16.71	16.00	12.62	13.58	14.73
Text-to-SPARQL	24.92	25.00	24.81	24.94	24.56	24.84
Text-to-Cypher	19.92	20.42	20.83	20.42	19.08	20.13
KB Routing	64.88	68.21	60.40	54.83	59.93	61.65
OmniRetrieval (Ours)	**68.58**	**73.30**	**66.47**	**57.81**	**62.40**	**65.71**
Oracle	*100.00*	*100.00*	*100.00*	*100.00*	*100.00*	*100.00*
Retrieval Accuracy						
Document Search	13.42	14.94	13.69	13.23	13.16	13.69
Text-to-SQL	13.51	16.75	15.46	12.78	13.92	14.48
Text-to-SPARQL	18.26	20.71	15.60	16.65	17.93	17.83
Text-to-Cypher	18.06	17.17	18.68	18.58	17.14	17.93
KB Routing	42.07	46.83	38.59	34.28	38.13	39.98
OmniRetrieval (Ours)	**46.62**	**52.69**	**43.07**	**38.34**	**40.97**	**44.34**
Oracle	*62.47*	*65.56*	*60.27*	*60.11*	*60.85*	*61.85*
LLM-as-a-Judge						
Document Search	39.93	40.92	40.82	36.31	39.47	39.49
Text-to-SQL	25.61	25.86	29.63	22.00	25.16	25.65
Text-to-SPARQL	28.89	30.29	31.06	23.99	25.70	27.99
Text-to-Cypher	33.09	24.36	34.85	24.42	25.19	28.38
KB Routing	60.26	63.67	61.93	50.37	53.71	57.99
OmniRetrieval (Ours)	**69.72**	**71.10**	**68.62**	**60.83**	**59.13**	**65.88**
Oracle	*75.20*	*76.50*	*76.29*	*71.93*	*72.81*	*74.55*

Table 1: Main results, with each metric macro-averaged across the four retrieval paradigms. Best results among the comparable methods are **bolded**; second best are underlined. *Oracle* is a upper bound with perfect source selection.

Oracle As a non-comparable upper bound, this method achieves perfect source selection, using the gold knowledge base annotated in each sample.

A direct comparison against methods that collapse heterogeneous sources into a unified representation (Oguz et al., 2022; Ma et al., 2022; Baek et al., 2023) is not realizable, since materializing such a representation is already infeasible at our benchmark scale, itself a small slice of real-world deployments. For example, while Wikipedia underlies several of our corpora at 7 million passages, Wikidata holds well over 15 billion triples[3], several orders of magnitude beyond typical dense indexes. The same gap recurs for labeled property graphs, where paths (the natural retrieval unit) grow exponentially with hop length and three-hop paths on one graph in our pool already reach tens of billions; and for relational databases, where one database in our pool holds over 70 million rows while row-level encoding further discards joins and set operations SQL is meant to express. Nonetheless, we report these methods under a constrained setup in Table 3.

4.3 Evaluation Metrics

We report results with the following three metrics.

Source Selection Accuracy It measures how often a method selects and includes both the correct backend and knowledge base for each question.

[3] https://www.wikidata.org/

Retrieval Accuracy It evaluates the quality of retrieved results: **NDCG@10** for document search (how well the retrieved ranking matches the gold relevance annotations), and **Execution Match** for SQL, SPARQL, and Cypher (whether the executed result set matches that of the gold query).

LLM-as-a-Judge The metrics above are strict: they require the predicted output to match the gold reference exactly, penalizing both the surface-form differences (such as a passage versus a table row) and routing to a legitimately alternative source. To complement them with a softer signal that tolerates these surface differences, we use an **LLM-as-a-Judge** (Liu et al., 2023) with GPT-5.4-mini (OpenAI, 2026): the judge sees the question, the prediction, and the gold annotation, and credits the prediction whenever the predicted output is semantically equivalent to gold or faithfully realizes the question against an alternative knowledge base.

4.4 Implementation Details

For each comparison, we instantiate all methods with the same backbone, which spans GPT-5.4 (OpenAI, 2026), Gemini-3.1 (Pro) (The Gemini Team, 2026), Sonnet-4.6 (Anthropic, 2026), Qwen-3.5 (27B) (Qwen Team, 2026), and Gemma-4 (31B) (Farabet and Lacombe, 2026), where closed-source backbones are served through their APIs, while open-source ones are served locally with

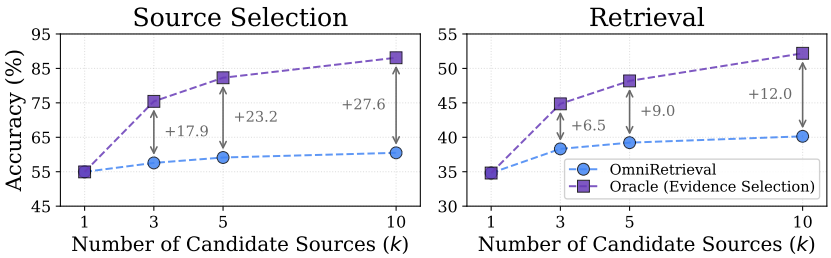

Figure 2: Effect of the candidate list size k in source selection on Source Selection and Retrieval accuracy. **Oracle (Evidence Selection)** replaces the LLM-based evidence selection with gold-source selection within the top-k candidates.

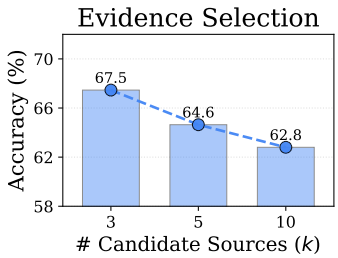

Figure 3: Evidence-selection accuracy on multi-candidate questions with the gold in the top-k.

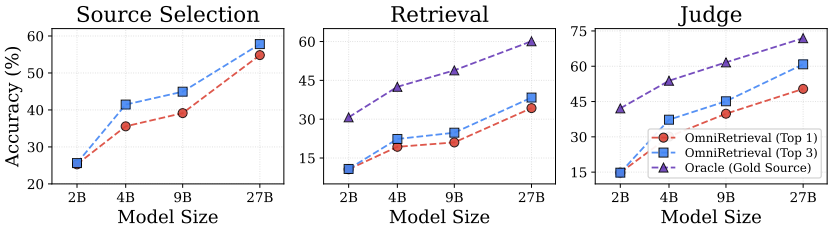

Figure 4: Effect of backbone scale (Qwen-3.5, 2B to 27B). **Oracle (Gold Source)** uses the gold source in place of LLM-based selection, and is omitted from Source Selection (where it is trivially 100% by construction).

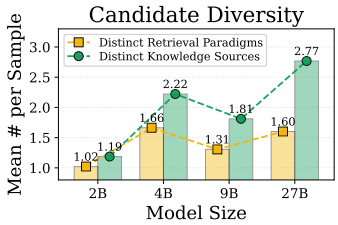

Figure 5: Candidate diversity: distinct retrieval paradigms and knowledge sources per sample.

vLLM (Kwon et al., 2023). For document retrieval, we use `all-MiniLM-L6-v2` (Reimers and Gurevych, 2019), but, instead of embedding the question directly, it is first rewritten into a hypothetical passage and then embedded. For text-to-SPARQL, the entity-linking step follows the procedure from Sun et al. (2024), used to build the context c_b. Further implementation details are in Appendix B, and the prompts are in Appendix E.

5 Experimental Results and Analyses

Main Results We report the main results in Table 1, where OmniRetrieval consistently outperforms all baselines across the five backbones. The four single-backend baselines, each pinned to one paradigm, perform poorly since three of the four query types lie outside their reach. KB Routing lifts this restriction by routing to one source per query, but its up-front commitment leaves no fallback when the chosen source is wrong. OmniRetrieval instead engages multiple candidates and consolidates them via cross-source evidence selection, yielding consistent gains over KB Routing. Also, the gap to the Oracle upper bound narrows from selection to judge (34.27 $\rightarrow$ 17.51 $\rightarrow$ 8.67 pts), suggesting that evidence selection often recovers a semantically equivalent answer from an alternative source even when source selection misses.

Analysis on Source Candidate Size To see the effect of candidate list size, we sweep top-k with Qwen-3.5 (27B) and additionally compare against an **Oracle (Evidence Selection)** variant that replaces the LLM-based selector with gold-source selection within the k. As shown in Figure 2, OmniRetrieval improves monotonically with k, yet the oracle improves much faster, widening the gap to the oracle as k grows. This observation traces to the selector itself: its multi-candidate 1-of-k accuracy drops from 67.5% at k=3 to 62.8% at k=10 (Figure 3). Thus, combined with the linear cost of additional candidates, this points to the initial evidence selection as the more impactful lever.

Analysis on Backbone Scale We next sweep the backbone scale, comparing OmniRetrieval (Top 1, one candidate per question) with OmniRetrieval (Top 3, our default). As shown in Figure 4, the two are essentially tied at 2B and Top 3 takes a clear lead at 27B, a gap that traces to candidate diversity (Figure 5): at 2B the source-selection step collapses to a single paradigm, while beyond 4B it produces meaningfully different candidates across both paradigms and sources[4]. Yet the gap to Oracle (Gold Source) ceiling remains, the largest on Source Selection, underscoring source selection as the most consequential step in the pipeline. This aligns with the design rationale of OmniRetrieval: broad exploration at source-selection, with the final commitment deferred to the evidence-selection.

[4]Qwen-3.5 9B exhibits a bias toward Document Search at its top candidate, pulling its candidates into the same retrieval paradigm; this surfaces as the diversity dip in Figure 5.

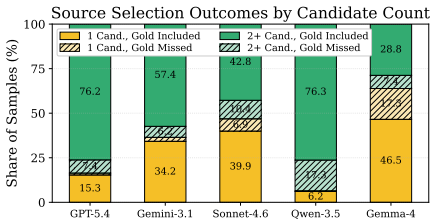

Figure 6: Source-selection behavior on 1 and 2+ candidate regimes. Solid segments mark success (gold present in the candidate list); hatched ones mark failure.

Backbone	Acc. (%)	Rand. (%)	Δ (pp)
GPT-5.4	72.81	38.31	+34.51
Gemini-3.1	75.29	43.99	+31.30
Sonnet-4.6	70.44	41.47	+28.98
Qwen-3.5	67.91	35.55	+32.36
Gemma-4	74.33	47.73	+26.60

Table 2: Evidence-selection accuracy on multi-candidate questions containing the gold. *Random* is the per-backbone uniform-pick baseline.

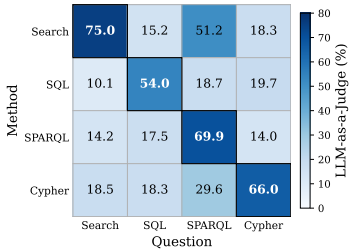

Figure 7: LLM-Judge accuracy on GPT-5.4 (rows: method paradigm; columns: question paradigm).

Method	Source Sel.	Retrieval	Judge
Document Search	21.49	13.42	39.93
Unified Representation†	31.00	23.00	45.00
KB Routing	64.88	42.07	60.26
OmniRetrieval (Ours)	**68.58**	**46.62**	**69.72**

Table 3: Results for the unified-representation methods (Oguz et al., 2022; Ma et al., 2022) with the constrained setup on GPT-5.4, marked † as non-comparable.

Analysis on Cross-Source Evidence Selection Building on the deferral argument above, we examine how reliably evidence selection commits to the gold candidates. The source-selection step lands in one of two regimes per question, either committing to a single candidate or emitting two or more, and the exploration versus commitment trade-off varies widely across backbones (Figure 6). Yet two patterns hold consistently. First, the gold source is included in the candidate list at a high rate at every backbone (Figure 6, solid segments). Second, once included, the evidence-selection step picks it at a high rate as well, far above the per-backbone random baseline (Table 2). Broad upstream exploration thus translates into accurate final commitments at the step that follows.

Analysis on Cross-Paradigm Coverage To see how much of the question space each paradigm can cover, Figure 7 reports a matrix of LLM-as-a-Judge accuracy on GPT-5.4, with rows as the method paradigm and columns as the question paradigm. From this, we find that Document Search has by far the widest cross-paradigm coverage, with an off-diagonal mean of 28.2% against 15.2 to 22.1% for the three structured paradigms. The advantage is largely driven by SPARQL questions, where the Wikipedia-derived corpora exposed under Search overlap with the factual content of Wikidata.

Analysis on Native vs Unified Retrieval To examine how shared-representation methods would perform once the pool is shrunk to a scale they can index (via sub-sampling), we build a constrained pool that retains only the gold-touched triples and edges in SPARQL and Cypher (with a same-count random distractor), includes all SQL tables, and keeps the document corpus at full scale. All items are verbalized and embedded into a single space with `all-MiniLM-L6-v2`, and the top-3 retrievals, matched to the candidate budget of OmniRetrieval, serve as the final results. As shown in Table 3, the unified method surpasses the four single-backend baselines through occasional cross-paradigm coverage, but stays far below KB Routing and OmniRetrieval. The gap persists even though the constrained setting hands this unified method an advantage no other method receives, pointing to a fundamental limit: atomic-unit retrieval cannot capture the structural composition (e.g., joins, traversals, multi-hop chains) that native queries express.

6 Conclusion

In this work, we presented OmniRetrieval, a framework for retrieval over structurally heterogeneous knowledge sources that, given a natural-language question, engages each relevant source through its own native query language and consolidates the executor outputs via a cross-source evidence selection step, rather than collapsing the sources into a shared representation. Evaluation on a benchmark spanning 13 datasets and 309 knowledge bases over unstructured corpora, relational databases, RDF graphs, and labeled property graphs shows that OmniRetrieval consistently outperforms relevant baselines. Our analyses further indicate that broad exploration at the source-selection step, with the final commitment deferred to a selector that rests on retrieved evidence, is what lets OmniRetrieval scale gracefully. These findings position OmniRetrieval as a step toward a general-purpose universal layer, one that preserves the structural affordances that make each source valuable while exposing a single natural-language interface to the user.

Limitations

OmniRetrieval is a general framework, and our specific instantiation in this work opens a couple of directions for further exploration. First, while our instantiation of the cross-source evidence selection step already performs reliably, it would be valuable to further strengthen this in future work, for example, through supervised fine-tuning on labeled cross-source selections or reinforcement learning that uses downstream answer quality as the reward signal. Second, our setup uses a single shared LLM across source selection, native query formulation, and evidence selection, and operator-specific specialization, while orthogonal to the framework, is another avenue worth investigating.

Ethical Considerations

Our work builds on publicly available datasets and standard LLMs, and we do not foresee ethical concerns beyond those inherited from these underlying components. In particular, since the retrieved evidence and the generated source-native queries draw on the connected knowledge bases and the internalized knowledge within the LLMs, any private, harmful, or biased content present in these resources may carry over to the outputs, and we recommend applying standard safeguards and filtering over both the connected sources and the generated queries to ensure safe and responsible deployment.

References

Anthropic. 2024. Introducing the next generation of claude.

Anthropic. 2026. Introducing Claude Sonnet 4.6. Anthropic Official Blog.

Jinheon Baek, Alham Fikri Aji, Jens Lehmann, and Sung Ju Hwang. 2023. Direct fact retrieval from knowledge graphs without entity linking. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 10038–10055. Association for Computational Linguistics.

Kurt Bollacker, Colin Evans, Praveen K. Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250. ACM.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Vera Boteva, Demian Gholipour Ghalandari, Artem Sokolov, and Stefan Riezler. 2016. A full-text learning to rank dataset for medical information retrieval. In *Advances in Information Retrieval - 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20-23, 2016. Proceedings*, Lecture Notes in Computer Science, pages 716–722. Springer.

Felix Brei, Lorenz Bühmann, Johannes Frey, Daniel Gerber, Lars-Peter Meyer, Claus Stadler, and Kirill Bulert. 2025. ARUQULA - an LLM based text2sparql approach using react and knowledge graph exploration utilities. In *Proceedings of the First International TEXT2SPARQL Challenge co-Located with Text2KG at ESWC25, Portorož, Slovenia, June 01, 2025*, CEUR Workshop Proceedings, pages 49–63. CEUR-WS.org.

Andrew Drozdov, Nathanael Schärli, Ekin Akyürek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, and Denny Zhou. 2023. Compositional semantic parsing with large language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. 2019. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*, Lecture Notes in Computer Science, pages 69–78. Springer.

Clement Farabet and Olivier Lacombe. 2026. Gemma 4: Byte for byte, the most capable open models. Google Official Blog.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2024. Text-to-sql empowered by large language models: A benchmark evaluation. *Proc. VLDB Endow.*, 17(5):1132–1145.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 1762–1777. Association for Computational Linguistics.

Gemini Team. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Jiayan Guo, Lun Du, and Hengyu Liu. 2023. Gpt4graph: Can large language models understand graph structured data ? an empirical evaluation and benchmarking. *arXiv preprint arXiv:2305.15066*.

Soyeong Jeong, Taehee Jung, Sung Ju Hwang, Joo-Kyung Kim, and Dongyeop Kang. 2025a. When thoughts meet facts: Reusable reasoning for long-context lms. *arXiv preprint arXiv:2510.07499*.

Soyeong Jeong, Kangsan Kim, Jinheon Baek, and Sung Ju Hwang. 2025b. Videorag: Retrieval-augmented generation over video corpus. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, Findings of ACL, pages 21278–21298. Association for Computational Linguistics.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023. Structgpt: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 9237–9251. Association for Computational Linguistics.

Karen Spärck Jones. 2004. A statistical interpretation of term specificity and its application in retrieval. *J. Documentation*, 60(5):493–502.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Jinhyuk Lee, Anthony Chen, Zhuyun Dai, Dheeru Dua, Devendra Singh Sachan, Michael Boratko, Yi Luan, Sébastien M. R. Arnold, Vincent Perot, Siddharth Dalmia, Hexiang Hu, Xudong Lin, Panupong Pasupat, Aida Amini, Jeremy R. Cole, Sebastian Riedel, Iftekhar Naim, Ming-Wei Chang, and Kelvin Guu. 2024. Can long-context language models subsume retrieval, rag, sql, and more? *arXiv preprint arXiv:2406.13121*.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chen-Chuan Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can LLM already serve as A database interface? A big bench for large-scale database grounded text-to-sqls. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: NLG evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 2511–2522. Association for Computational Linguistics.

Kaixin Ma, Hao Cheng, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. 2022. Open domain question answering with A unified knowledge interface. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 1605–1620. Association for Computational Linguistics.

Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. 2018. Www'18 open challenge: Financial opinion mining and question answering. In *Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018, Lyon , France, April 23-27, 2018*, pages 1941–1942. ACM.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, CEUR Workshop Proceedings. CEUR-WS.org.

Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Sejr Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. 2022. Unik-qa: Unified representations of structured and unstructured knowledge for open-domain question answering. In *Findings of the Association for Computational Linguistics: NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, Findings of ACL, pages 1535–1546. Association for Computational Linguistics.

OpenAI. 2026. Introducing GPT-5.4. OpenAI Official Blog.

OpenAI. 2026. Openai GPT-5 system card. *arXiv preprint arXiv:2601.03267*.

Makbule Gulcin Ozsoy, Leila Messallem, Jon Besga, and Gianandrea Minneci. 2025. Text2cypher: Bridging natural language and graph databases. In *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025 - Workshops, Abu Dhabi, UAE, January 19-24, 2025*, pages 100–108. Association for Computational Linguistics.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive apis. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Qwen Team. 2026. Qwen3.5: Towards native multimodal agents.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, Proceedings of Machine Learning Research, pages 8748–8763. PMLR.

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models. *arXiv preprint arXiv:2204.00498*.

Nils Reimers and Iryna Gurevych. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994*, NIST Special Publication, pages 109–126. National Institute of Standards and Technology (NIST).

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Phillip Schneider, Manuel Klettner, Kristiina Jokinen, Elena Simperl, and Florian Matthes. 2024. Evaluating large language models in semantic parsing for conversational question answering over knowledge graphs. In *Proceedings of the 16th International Conference on Agents and Artificial Intelligence, ICAART 2024, Volume 3, Rome, Italy, February 24-26, 2024*, pages 807–814. SCITEPRESS.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

The Gemini Team. 2026. Gemini 3.1 pro: A smarter model for your most complex tasks. Google Official Blog.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 809–819. Association for Computational Linguistics.

Ricardo Usbeck, Xi Yan, Aleksandr Perevalov, Longquan Jiang, Julius Schulz, Angelie Kraft, Cedric Möller, Junbo Huang, Jan Reineke, Axel-Cyrille Ngonga Ngomo, Muhammad Saleem, and Andreas Both. 2024. QALD-10 - the 10th challenge on question answering over linked data: Shifting from dbpedia to wikidata as a KG for KGQA. *Semantic Web*, 15(6):2193–2207.

Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or fiction: Verifying

scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 7534–7550. Association for Computational Linguistics.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Woongyeong Yeo, Kangsan Kim, Soyeong Jeong, Jinheon Baek, and Sung Ju Hwang. 2025. Universalrag: Retrieval-augmented generation over multiple corpora with diverse modalities and granularities. *arXiv preprint arXiv:2504.20734*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3911–3921. Association for Computational Linguistics.

A Benchmark Details

This section provides additional details on our benchmark, including example questions (Table 4), the verbatim structural context c_b for each backend, and a note on the use of existing artifacts.

A.1 Document Search

For each corpus, c_b is a short topical descriptor comprising a one-line description of the domain, and the typical document style. The example below is the verbatim c_b used for NFCorpus.

```
Corpus: nfcorpus
Description: Medical and nutritional
  information retrieval
Query type: a consumer-health or
  nutrition query, often very terse,
  either a one-to-three-word topic stub
  (a food, condition, or substance) or
  a popular-science headline phrased
  for a lay audience
Document style: a PubMed-style biomedical
  research abstract with a formal
  scientific title, written in technical
  register
```

A.2 Relational Databases

For each relational database, c_b is the schema, exposed as the `CREATE TABLE` declarations of all tables, including column types, primary keys, and foreign keys. The example below is the verbatim c_b for the Spider database `concert_singer` (one of four tables shown; the other three are omitted).

```
CREATE TABLE "singer" (
  "Singer_ID" int,
  "Name" text,
  "Country" text,
  "Song_Name" text,
  "Song_release_year" text,
  "Age" int,
  "Is_male" bool,
  PRIMARY KEY ("Singer_ID")
)
[CREATE TABLE statements for concert,
 singer_in_concert, and stadium omitted]
```

A.3 RDF Knowledge Graphs

For Wikidata, c_b is built per question and combines (i) a fixed preamble of SPARQL prefixes and schematic query templates with (ii) the topic entities mentioned in the question, linked to Wikidata QIDs, and (iii) the top-30 candidate predicates per entity, ranked by semantic similarity between the predicate label and the question. The example below is an abridged c_b for the LC-QuAD 2.0 question shown in Table 4.

```
Knowledge graph: Wikidata
Prefixes: wd: (entity),
  wdt: (direct/truthy property),
  p: (entity -> statement node),
  ps: (statement -> main value),
  pq: (statement -> qualifier value),
  rdfs: (for rdfs:label)
Format examples (schematic IDs; use the
  actual topic entities and relations
  for the query):
  SELECT ?x WHERE { wd:Qxxx wdt:Pyyy ?x }
  SELECT ?x WHERE { wd:Qxxx wdt:Pyyy ?y .
                    ?y wdt:Pzzz ?x }
  ASK WHERE { wd:Qxxx wdt:Pyyy wd:Qzzz }
  [count, filter, order-by forms omitted]

Topic entities (from the question,
  already linked to Wikidata QIDs):
- wd:Q188920  (Delta Air Lines)
- wd:Q1002697 (periodical)

Linked relations (candidate properties
  per topic entity, choose among these):
- wd:Q188920  (Delta Air Lines):
    - P31   (instance of)
    - P229  (IATA airline designator)
    - P2813 (house publication)
    [further candidates omitted]
- wd:Q1002697 (periodical):
    - P31  (instance of)
    [further candidates omitted]
```

A.4 Labeled Property Graphs

For each Text2Cypher graph, c_b is the graph schema, exposed as node labels with typed property keys, relationship types with typed property keys, and the list of edges as triples. The example below is the verbatim c_b for the database `neo4jlabs_demo_db_movies`.

```
Node properties:
- Movie
  - `title`: STRING Example: "The Matrix"
  - `votes`: INTEGER Min: 1, Max: 5259
  - `tagline`: STRING
    Example: "Welcome to the Real World"
  - `released`: INTEGER Min: 1975, Max: 2012
- Person
  - `born`: INTEGER Min: 1929, Max: 1996
  - `name`: STRING Example: "Keanu Reeves"
Relationship properties:
- ACTED_IN
  - `roles: LIST` Min Size: 1, Max Size: 6
- REVIEWED
  - `summary: STRING` Available options: [...]
  - `rating: INTEGER` Min: 45, Max: 100
The relationships:
(:Person)-[:ACTED_IN]->(:Movie)
(:Person)-[:DIRECTED]->(:Movie)
(:Person)-[:PRODUCED]->(:Movie)
(:Person)-[:WROTE]->(:Movie)
(:Person)-[:FOLLOWS]->(:Person)
(:Person)-[:REVIEWED]->(:Movie)
```

Backend	Dataset	Example Question
Document Search	NFCorpus	"Cancer Risk From French Fries"
	SciFact	"Myelin sheaths play a role in action potential propagation."
	FiQA	"Credit card closed. Effect on credit score?"
	MS MARCO	"where is wetransfer based"
	FEVER	"Sean Connery was cast in The Rock."
	Natural Questions	"who was originally offered the role in pretty woman"
	HotpotQA	"Which software application was originally made for the company that acquired RadioShack in 1963?"
Relational Databases	Spider	"What are all distinct countries where singers above age 20 are from?"
	BIRD	"How many online purchases did Ole Group make in May 2019?"
RDF Knowledge Graphs	SimpleQuestions	"Who discovered 20468 petercook?"
	QALD-10	"Which artists were born on the same date as Rachel Stevens?"
	LC-QuAD 2.0	"What periodical literature does Delta Air Lines use as a mouthpiece?"
Labeled Property Graphs	Text2Cypher	"Which actors have acted in movies directed by the person who directed 'Speed Racer'?"

Table 4: One example question per dataset, covering the 13 datasets that together span the four native backends.

A.5 Use of Existing Artifacts

All artifacts used in our benchmark (the 13 datasets and 309 knowledge bases) are existing public resources, and we use each under its respective license and terms. Our use is restricted to research purposes, consistent with the intended use specified by the original releases, and we do not perform additional filtering for personally identifying information or offensive content, deferring to the policies of the upstream datasets.

B Implementation Details

We use a sampling temperature of 0.0 across all LLM calls (source selection, query formulation, and cross-source evidence selection), which makes the predictions deterministic, so all reported numbers come from a single run per configuration. The maximum number of generated tokens is capped at 1024. Open-source backbones are served locally on a single NVIDIA H200 GPU.

C Per-Paradigm Breakdown

Table 5 reports the per-paradigm decomposition of the three metrics from Table 1, broken down by the four native retrieval paradigms (Search, SQL, SPARQL, Cypher) at each of the five backbones. The macro average across the four columns recovers the corresponding column in Table 1.

D Paradigm Prediction Balance

To examine how source selection allocates predictions across the four retrieval paradigms, we report the predicted paradigm distribution in Figure 8, under per-paradigm balanced weighting in which each gold paradigm contributes equally to the totals, so the reference becomes a uniform 25% line per paradigm. We show both the top-1 candidate and the full pool of emitted candidates. From this, we observe that the predicted distribution stays within a narrow band of the balanced reference across all five backbone models: SQL ranges over 30 to 37% at top-1, and Search ranges over 27 to 37%, indicating that source selection remains broadly balanced across paradigms even though the underlying catalog is dominated by relational databases ($\approx 93\%$, 286 of 309 knowledge sources).

E Prompts

This section lists the prompts used in our framework, covering the source selection, query formulation, and cross-source evidence selection steps, as well as the LLM-as-a-Judge metric.

Source Selection Figure 9 shows the prompt used for source selection. Given a catalog of available KBs grouped by backend, where each KB is summarized by a short descriptor (description, examples, or schema summary), the model returns up to k ranked candidate sources for the question.

Query Formulation Figure 10 shows the system prompts for the three native query languages and the shared user template that grounds the query in the backend context c_b together with the question. For document retrieval, the question is additionally rewritten into a hypothetical passage with the prompt in Figure 11, similar to Gao et al. (2023).

Cross-Source Evidence Selection Figure 12 shows the prompt for picking the best candidates among multiple source-specific retrieval results.

LLM-as-a-Judge Figure 13 shows the prompt used by the LLM-as-a-Judge metric, used to assess whether the predicted answer answers the question by direct match or faithful realization against gold and question on an alternative knowledge base.

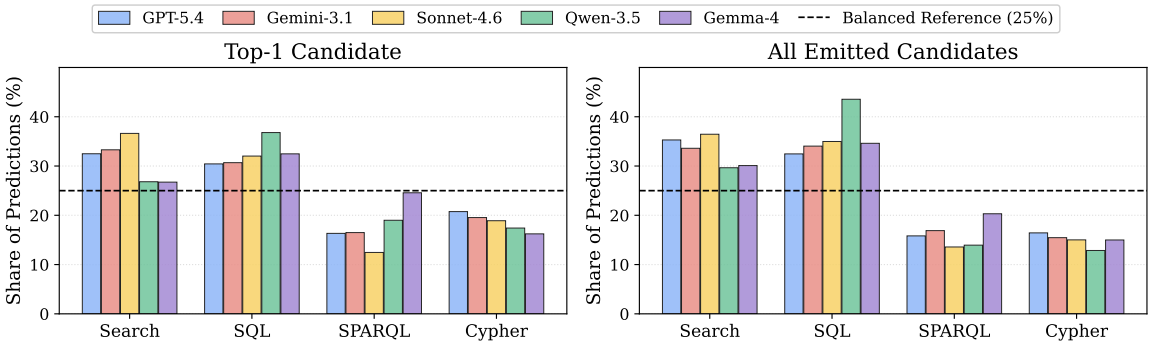

Figure 8: Predicted retrieval paradigm distribution under per-paradigm balanced weighting. **Left:** top-1 candidate per question. **Right:** all emitted candidates. The dashed line marks the uniform 25% reference.

Backbone	**Method**	**Search** (R / E / J)	**SQL** (R / E / J)	**SPARQL** (R / E / J)	**Cypher** (R / E / J)
GPT-5.4	Document Search	85.95 / 48.62 / 75.00	0.00 / 2.17 / 15.17	0.00 / 0.22 / 51.22	0.00 / 2.67 / 18.33
	Text-to-SQL	0.00 / 0.00 / 10.10	59.00 / 42.50 / 54.00	0.00 / 5.89 / 18.67	0.00 / 5.67 / 19.67
	Text-to-SPARQL	0.00 / 0.00 / 14.19	0.00 / 7.83 / 17.50	99.67 / 60.56 / 69.89	0.00 / 4.67 / 14.00
	Text-to-Cypher	0.00 / 0.00 / 18.48	0.00 / 8.67 / 18.33	0.00 / 6.89 / 29.56	79.67 / 56.67 / 66.00
	KB Routing	78.29 / 44.02 / 70.81	57.00 / 40.17 / 51.67	55.22 / 36.11 / 56.89	69.00 / 48.00 / 61.67
	OmniRetrieval (Ours)	**75.33 / 44.09 / 78.14**	**61.00 / 41.50 / 56.17**	**66.67 / 51.22 / 77.22**	**71.33 / 49.67 / 67.33**
	Oracle	*100.00 / 54.51 / 79.19*	*100.00 / 63.83 / 72.50*	*100.00 / 60.89 / 69.78*	*100.00 / 70.67 / 79.33*
Gemini-3.1 (Pro)	Document Search	90.43 / 52.15 / 78.33	0.00 / 2.83 / 13.67	0.00 / 0.78 / 52.00	0.00 / 4.00 / 19.67
	Text-to-SQL	0.00 / 0.00 / 9.62	66.83 / 52.67 / 59.50	0.00 / 6.67 / 17.67	0.00 / 7.67 / 16.67
	Text-to-SPARQL	0.00 / 0.00 / 15.38	0.00 / 7.50 / 12.67	100.00 / 71.67 / 79.11	0.00 / 3.67 / 14.00
	Text-to-Cypher	0.00 / 0.00 / 10.67	0.00 / 8.67 / 9.33	0.00 / 7.00 / 16.11	81.67 / 53.00 / 61.33
	KB Routing	87.33 / 50.94 / 78.14	64.50 / 50.17 / 60.00	56.67 / 40.56 / 63.56	64.33 / 45.67 / 53.00
	OmniRetrieval (Ours)	**81.19 / 48.65 / 81.52**	**68.67 / 52.33 / 63.67**	**78.67 / 62.11 / 79.89**	**64.67 / 47.67 / 59.33**
	Oracle	*100.00 / 56.13 / 80.24*	*100.00 / 70.00 / 73.67*	*100.00 / 71.44 / 79.78*	*100.00 / 64.67 / 72.33*
Sonnet-4.6	Document Search	87.62 / 48.33 / 77.43	0.00 / 2.33 / 15.17	0.00 / 0.11 / 51.33	0.00 / 4.00 / 19.33
	Text-to-SQL	0.00 / 0.00 / 17.95	64.00 / 47.83 / 58.67	0.00 / 6.33 / 22.22	0.00 / 7.67 / 19.67
	Text-to-SPARQL	0.00 / 0.00 / 20.52	0.00 / 8.50 / 18.17	99.22 / 48.89 / 68.56	0.00 / 5.00 / 17.00
	Text-to-Cypher	0.00 / 0.00 / 22.90	0.00 / 9.17 / 18.17	0.00 / 6.89 / 29.00	83.33 / 58.67 / 69.33
	KB Routing	77.81 / 42.62 / 72.95	60.33 / 44.50 / 56.33	40.44 / 19.89 / 58.78	63.00 / 47.33 / 59.67
	OmniRetrieval (Ours)	**76.10 / 42.35 / 77.38**	**63.33 / 46.17 / 60.00**	**62.44 / 37.11 / 73.44**	**64.00 / 46.67 / 63.67**
	Oracle	*100.00 / 53.65 / 79.43*	*100.00 / 66.33 / 75.17*	*100.00 / 49.44 / 68.89*	*100.00 / 71.67 / 81.67*
Qwen-3.5 (27B)	Document Search	85.95 / 44.41 / 71.62	0.00 / 3.17 / 12.50	0.00 / 1.00 / 45.44	0.00 / 4.33 / 15.67
	Text-to-SQL	0.00 / 0.00 / 9.43	50.50 / 38.00 / 46.67	0.00 / 5.78 / 15.56	0.00 / 7.33 / 16.33
	Text-to-SPARQL	0.00 / 0.00 / 8.29	0.00 / 8.83 / 10.00	99.78 / 52.78 / 62.67	0.00 / 5.00 / 15.00
	Text-to-Cypher	0.00 / 0.00 / 8.62	0.00 / 8.33 / 8.50	0.00 / 6.67 / 13.22	81.67 / 59.33 / 67.33
	KB Routing	66.38 / 34.45 / 58.24	45.17 / 34.33 / 42.67	57.78 / 33.00 / 53.56	50.00 / 35.33 / 47.00
	OmniRetrieval (Ours)	**56.14 / 31.12 / 67.86**	**56.00 / 39.67 / 50.33**	**64.78 / 42.22 / 68.44**	**54.33 / 40.33 / 56.67**
	Oracle	*100.00 / 50.72 / 77.00*	*100.00 / 64.17 / 69.50*	*100.00 / 52.89 / 62.56*	*100.00 / 72.67 / 78.67*
Gemma-4 (31B)	Document Search	85.67 / 44.30 / 72.62	0.00 / 3.33 / 17.17	0.00 / 0.67 / 47.44	0.00 / 4.33 / 20.67
	Text-to-SQL	0.00 / 0.00 / 12.90	54.33 / 42.67 / 49.83	0.00 / 6.33 / 24.56	0.00 / 6.67 / 13.33
	Text-to-SPARQL	0.00 / 0.00 / 10.90	0.00 / 9.50 / 10.33	98.22 / 57.89 / 65.56	0.00 / 4.33 / 16.00
	Text-to-Cypher	0.00 / 0.00 / 13.52	0.00 / 9.33 / 10.00	0.00 / 6.89 / 16.56	76.33 / 52.33 / 60.67
	KB Routing	67.95 / 34.90 / 63.62	47.67 / 38.50 / 46.67	69.78 / 43.11 / 59.56	54.33 / 36.00 / 45.00
	OmniRetrieval (Ours)	**75.00 / 39.38 / 71.48**	**50.50 / 40.83 / 49.83**	**66.44 / 45.00 / 64.56**	**57.67 / 38.67 / 50.67**
	Oracle	*100.00 / 50.19 / 77.48*	*100.00 / 67.33 / 72.00*	*100.00 / 57.89 / 65.78*	*100.00 / 68.00 / 76.00*

Table 5: Per-paradigm decomposition of the three metrics from Table 1, broken down by the four native retrieval paradigms (Search, SQL, SPARQL, Cypher) across the five backbones. Each cell is reported as "R / E / J", where R is Source Selection Accuracy (both the backend and knowledge base correct), E is Retrieval Accuracy (NDCG@10 for Search, Execution Match for the structured backends), and J is LLM-as-a-Judge accuracy.

```
[System]
You are a query router. Given a question, decide which backend to use (SEARCH, SQL, SPARQL, or
CYPHER) and which knowledge base to query. Some queries are ambiguous and may match multiple
knowledge bases, return up to <k> routing decisions, most likely first; return fewer if you are
confident.
```

```
[User]


Available knowledge bases:

  SEARCH:
    - <kb_id> [<description> | query type: <...> | examples: <...>]
    - ...
  SQL:
    - <kb_id> [<table names>]
    - ...
  SPARQL:
    - <kb_id> [<description> | examples: <...>]
    - ...
  CYPHER:
    - <kb_id> [nodes: <...> | rels: <...>]
    - ...

Question: <question>

Respond with JSON: {"decisions": [{"route_type": "...", "kb_id": "..."}, ...]}
```

Figure 9: Source selection prompt.

```
[System for Text-to-SQL]
You are a text-to-SQL translator. Output only the SQL query.
[System for Text-to-SPARQL]
You are a text-to-SPARQL translator. Output only the SPARQL query.
[System for Text-to-Cypher]
You are a text-to-Cypher translator. Output only the Cypher query.
```

```
[User, shared across backends]


<c_b>

Question: <question>

Generate the <SQL|SPARQL|CYPHER> query.
```

Figure 10: Query formulation prompts for the structured backends.

[System for Document Search]

You are a search query optimizer for a dense retriever. Given the user query and a description of the target corpus, write a hypothetical passage that would be relevant evidence for the query, written in the register, style, and approximate length of documents in that corpus. The passage will be embedded and matched against real corpus documents, so favor concrete, in-domain content over generic phrasing.
Begin your output with the user query verbatim on its own line, just the query text, not the 'Question:' label that precedes it, then on the next line write the hypothetical passage. This keeps the literal query terms in the embedding alongside the semantic expansion.
If the query is a short topic stub or short keyword-style query (just a handful of words, often lowercase and without punctuation), do NOT write a passage, output the verbatim query and nothing else. The bare term already gives a strong dense-retrieval signal, and a hallucinated passage tends to lock onto one specific aspect that may not match the gold document.
Output only the verbatim query followed by the passage (or for a stub query, only the verbatim query), no preamble, no quotes, no labels.

Figure 11: Query rewriting prompt used in document retrieval. The user prompt template here is the same as for the other backends, shown in Figure 10.

[System]

You are a result selector. Pick the candidate whose result best answers the question.

[User]

Question: <question>

Candidates (each prefixed with its integer index in brackets, e.g. [0], [1], [2]):

[0] <route_type> | <kb_id>
query: <query>
<context and result>

[1] <route_type> | <kb_id>
query: <query>
<context and result>

...

Respond with JSON: {"selected": <integer index>}

Figure 12: Cross-source evidence selection prompt.

[System]

You are a strict but fair evaluator. You will see a user question and two sides: a PREDICTED side (the model's chosen KB) and a GOLD side (the labeled KB, known to be correct). Each side carries its KB schema/context, the query that was run, and the resulting answer.
Decide whether the predicted side correctly answers the user question. There are two independent ways the prediction can be correct: (1) ANSWER MATCH, the predicted answer is equivalent in meaning to the gold answer, allowing reordering, alias differences, formatting differences, or extra surrounding context; (2) FAITHFUL IMPLEMENTATION ON A DIFFERENT KB, the predicted query faithfully realizes what the user asked, interpreted against the predicted KB schema and data, and the predicted answer is what that query correctly produces. The values may differ entirely from gold because the predicted KB legitimately holds different content. This case applies whenever more than one knowledge base could reasonably answer the same kind of question.
If the gold answer is empty or otherwise degenerate (the gold query may itself be buggy or stale), the gold reference is uninformative, judge by FAITHFUL IMPLEMENTATION alone in that case. Reject when the predicted answer is off-topic or when the predicted query plainly fails to capture what the question is asking. If both pred and gold answers are empty, ALWAYS count it as an ANSWER MATCH (this overrides any reasoning about whether the question should have a real-world answer). If only the predicted answer is empty, reject. Use the schemas and queries on both sides to interpret unfamiliar values.

[User]

Question: <question>

[PREDICTED] route=<route> | kb=<kb_id>
query: <query>
<context and answer>

[GOLD] route=<route> | kb=<kb_id>
query: <query>
<context and answer>

Respond with JSON: {"correct": true|false, "reason": "<one-line reason>"}

Figure 13: LLM-as-a-Judge prompt used to assess answer correctness.