A Reference Architecture for Agentic Hybrid Retrieval in Dataset Search

Riccardo Terrenzi
Centre for Industrial Software
University of Southern Denmark
Alsion 2, Sønderborg, 6400, Denmark
rite@mmmi.sdu.dk

Phongsakon Mark Konrad
Centre for Industrial Software
University of Southern Denmark
Alsion 2, Sønderborg, 6400, Denmark
phkon23@student.sdu.dk

Tim Lukas Adam
Centre for Industrial Software
University of Southern Denmark
Alsion 2, Sønderborg, 6400, Denmark
tiada23@student.sdu.dk

Serkan Ayvaz
Centre for Industrial Software
University of Southern Denmark
Alsion 2, Sønderborg, 6400, Denmark
seay@mmmi.sdu.dk

***Abstract*—Ad hoc dataset search requires matching underspecified natural-language queries against sparse, heterogeneous metadata records, a task where typical lexical or dense retrieval alone falls short. We reposition dataset search as a software-architecture problem and propose a bounded, auditable reference architecture for agentic hybrid retrieval that combines BM25 lexical search with dense-embedding retrieval via reciprocal rank fusion (RRF), orchestrated by a large language model (LLM) agent that repeatedly plans queries, evaluates the sufficiency of results, and reranks candidates. To reduce the vocabulary mismatch between user intent and provider-authored metadata, we introduce an offline metadata augmentation step in which an LLM generates pseudo-queries for each dataset record, augmenting both retrieval indexes before query time. Two architectural styles are examined: a single ReAct agent and a multi-agent horizontal architecture with Feedback Control. Their quality-attribute tradeoffs are analyzed with respect to modifiability, observability, performance, and governance. An evaluation framework comprising seven system variants is defined to isolate the contribution of each architectural decision. The architecture is presented as an extensible reference design for the software architecture community, incorporating explicit governance tactics to bound and audit nondeterministic LLM components.**

***Index Terms*—dataset search, hybrid retrieval, software architecture, LLM agents, metadata augmentation, reference architecture**

I. Introduction and Motivation

Although open data portals contain millions of dataset records, identifying a dataset that precisely addresses a specific research question remains challenging. For example, a query for "COVID hospitalization rates by county" may yield no results for a dataset titled "US Health Statistics 2020, County-Level Hospital Admissions." This persistent barrier stems from a mismatch between user intent and provider-authored metadata [1]. In addition to vocabulary mismatch, dataset metadata frequently lacks detail: titles are often terse, descriptions are incomplete, and tags are inconsistent across different portals [2].

These issues are not merely usability problems: they systematically degrade retrieval quality in settings where the correct dataset is present but not lexically discoverable under the user's query. Traditional dataset search pipelines typically assume a single-pass formulation (query $\rightarrow$ retrieve $\rightarrow$ rank) [3], placing the burden of query refinement on the user and relying on portal-specific indexing choices. However, dataset search is inherently iterative and multi-faceted: users often need to reconcile synonyms, units, temporal and geographic scope, population definitions, and licensing constraints before a dataset can be judged relevant [4]. When metadata is sparse, even strong neural retrieval models may fail because the evidence required to establish relevance is simply absent from the indexed record [3]–[5].

Current dataset search engines attempt to solve these challenges at the system level. Google Dataset Search [1] aggregates schema.org[1] markup across the web, while Auctus [2] combines profiling with keyword search. However, neither system incorporates iterative retrieval procedures or semantic matching techniques.

The rise of LLM-based agents gives the opportunity to shift dataset search from a single-pass ranking problem to an iterative, tool-orchestrated workflow in which the system plans, retrieves, evaluates evidence, and revises queries over multiple steps [6] [7]. This shift is primarily architectural: it introduces a control loop, tool boundaries, and new governance requirements (e.g., bounded cost/latency, observability, reproducibility, and auditability).

To this end, we introduce an agent-based reference architecture for dataset search. The proposed architecture elevates retrieval from a single-stage function to a bounded, iterative workflow structured around a Plan–Retrieve–Evaluate loop. An LLM-based agent decomposes the user request, orchestrates hybrid retrieval (sparse and dense) through reciprocal rank fusion (RRF), evaluates intermediate results, and selectively rewrites queries under an explicit iteration budget. This control loop makes refinement systematic rather than

[1]https://schema.org

user-dependent, while preserving observability and bounded execution.

Beyond the single-agent design, we further propose (i) an offline metadata augmentation strategy based on LLM-generated pseudo-queries to mitigate first-pass retrieval failures caused by incomplete metadata, and (ii) a multi-agent decomposition in which specialized agents interact through typed contracts under feedback control, enabling higher performances, clearer responsibility boundaries and improved governability.

This paper offers three primary contributions:

1) A single-agent reference architecture for dataset search that fuses sparse and dense retrieval via reciprocal rank fusion (RRF). The architecture is governed by an LLM agent with an iterative Plan–Retrieve–Evaluate loop (Section III).
2) An offline metadata augmentation strategy using LLM-generated pseudo-queries to reduce first-pass retrieval failures (Section IV).
3) A multi-agent decomposition, implemented as a specialized agent pipeline with a feedback loop, together with typed inter-agent contracts. We also provide a quality-attribute tradeoff analysis and a governance checklist for bounding, auditing, and reproducing LLM-driven retrieval (Sections V and VI).

Our hypotheses are: (i) the single-agent architecture outperforms retrieval-only baselines in terms of ranking metrics (i.e. NDCG), demonstrating that iterative architectural control yields measurable gains over static pipelines; (ii) the offline pseudo-query augmentation reduces first-pass retrieval failures and decreases the number of iterations required to reach high-quality results; (iii) the multi-agent decomposition exceeds the retrieval performances of the single-agent architecture and improves modularity, observability, failure isolation, and governance.

The objective of this study is to reposition dataset search as an architectural design problem rather than solely an information retrieval problem, and provide a reproducible reference architecture for building bounded, auditable, and evolvable LLM-driven retrieval systems.

II. Related Work

1) Dataset Search and Metadata-Centric Retrieval: Recent dataset discovery work increasingly treats metadata as the main retrieval substrate, driven by scalability and access constraints. Paton et al. survey the dataset discovery design space and show that, even at portal scale, search quality is often limited more by sparse, heterogeneous metadata than by corpus size [4]. Human-in-the-loop evidence likewise shows persistent failures from vocabulary mismatch, ambiguous intent, and missing metadata context, motivating iterative, interaction-aware retrieval over single-shot keyword search [8]. Recent systems address this with richer metadata-centric representations: Metam models goal-oriented discovery by operationalizing user goals through intermediate reasoning over dataset descriptors and candidate transformations [9], while BLEND frames data discovery as an end-to-end pipeline unifying tasks such as entity- and attribute-centric lookup, again stressing robust metadata acquisition and integration as prerequisites for effective search [10].

2) Hybrid Retrieval and Retrieval-Augmented Generation: Recent RAG surveys describe a shift from "naïve" pipelines to modular designs that separately optimize retrieval, evidence selection, and generation [11]. On the retrieval side, sparse (lexical) and dense (semantic) signals are widely viewed as complementary, motivating fusion strategies such as RFF. Yang et al. propose sparse-guided partial dense retrieval, which evaluates only a subset of embedding clusters to preserve fusion quality while lowering dense retrieval cost [12]. A second line of work improves retrieval through query adaptation and diversification: Rewrite–Retrieve–Read casts RAG as query rewriting and shows that reformulating the input can substantially improve downstream results [13], while RAG-Fusion generates multiple queries and applies reciprocal-rank-style fusion to improve recall and coverage under paraphrase and ambiguity [14]. A third line adds control policies for when and how to retrieve: Self-RAG learns retrieval-on-demand and uses self-critique signals to regulate evidence use during generation [15].

3) Agentic Architectures and LLM-Orchestrated Systems: Recent work treats LLM agents as compound software systems whose reliability depends on orchestration, interfaces, and governance. The Agent Design Pattern Catalogue synthesizes recurring patterns and tradeoffs for foundation-model-based agents, enabling more principled decomposition and observability [16]. Moving from guidance to evaluation, AgentArcEval adapts scenario-based architecture assessment to FM-based agents and targets quality attributes that model-only metrics often miss, such as autonomy and continuous evolution [17]. Multi-agent orchestration is studied for both capability and engineering control: Mixture-of-Agents formalizes layered collaboration across multiple LLMs [18], while Gradientsys introduces a scheduler that routes tasks among specialized agents via typed interfaces and explicit replanning under failure [19]. Governance is also becoming runtime infrastructure; MI9 emphasizes continuous monitoring and conformance checks for agentic behavior beyond pre-deployment controls [20]. Finally, interactive benchmarks make architectural claims testable: X-WebAgentBench evaluates multilingual web-agent planning and interaction [21], and LoCoBench-Agent extends long-context software engineering evaluation to multi-turn, tool-using workflows with efficiency and recovery metrics [22].

III. Approach: Agentic Hybrid Retrieval

We address the problem of ad-hoc dataset search for tabular data. Given a collection of tables $\mathcal{D} = \{D_1, \ldots, D_M\}$ and a natural language query q, the goal is to return a ranked list of relevant datasets.

We frame the proposed architecture as a single-agent, iterative hybrid RAG architecture with a Plan-Retrieve-Evaluate control loop: a single LLM agent controls a hybrid retriever

over dataset metadata and performs listwise relevance reasoning over the retrieved candidates. The proposed architecture is shown in Fig. 1.

Moreover, we adopt dataset metadata as the primary searchable representation, since most dataset search in open data portals operates over publisher-provided metadata and typically expose keyword search and faceted metadata filtering rather than direct content-based search [1], [3], [4]. This setting is particularly challenging because metadata is often sparse, inconsistently defined across portals, and authored without a shared vocabulary, amplifying ambiguity and vocabulary mismatch at query time [3], [4].

At a high level, the architecture consists of a single LLM agent, two complementary metadata stores and one retrieval tool:

- Single LLM agent: an agent responsible for query planning, retrieval, candidates evaluation and candidates listwise reranking.
- Lexical metadata index: an inverted index built over serialized dataset metadata, supporting classical lexical matching (BM25).
- Vector database: a vector store containing dense embeddings of the same serialized metadata, enabling semantic retrieval.
- Retriever tool: executes lexical and dense retrieval and returns a fused candidate set with provenance.

Concretely, instead of delegating query planning, retrieval evaluation and reranking to separate autonomous components, the single agent performs these steps autonomously within its own decision loop, conditioning on the retrieved observations. This design choice is motivated by recent work showing that agentic RAG-style controllers can improve robustness by iterating between retrieval and self-critique, rather than relying on a single retrieve-then-rank step [23]–[25]. Moreover, in the dataset domain, benchmark collections such as the NTCIR-derived ad hoc dataset retrieval test collection [26] and ACORDAR 2.0 [27] emphasize that successful dataset retrieval often requires semantic interpretation beyond simple term matching. This evidence supports the decision to use a hybrid, iterative retrieval phase.

A. Data Stores population

The metadata catalogue content is used to populate both a BM25 inverted index and a VectorDB. It is usually composed of entries like this one:

```
{
  {
    "download": "...",
    "size": "...",
    "author": "...",
    "created": "...",
    "dataset_id": "...",
    "description": "...",
    "title": "...",
    "version": "...",
    "tags": "..."
  },
  ...
}
```

In order to populate the inverted index and the vector database, each dataset record is serialized as a flat text string with the following fields. We select these fields because they carry most of the lexical and semantic evidence available for content-based dataset search: titles and descriptions capture topical meaning and paraphrases, tags provide high-precision keywords, and provider/author fields often include distinctive named entities that help disambiguate similar datasets.

```
Title is {...}, Description is {...},
Tags are {...}, Author is {...}
```

B. Single agent workflow

The workflow begins when the system receives a natural-language user query. The agent will follow a Plan–Retrieve–Evaluate cycle, inspired by ReAct-style reasoning [28]. The agent first analyzes the query and decides how to optimize it for retrieval, e.g., by decomposing it into sub-questions, diversifying it into multiple complementary formulations, and determining whether a rewrite is necessary to reduce vocabulary mismatch. Based on this analysis, the agent builds a query plan specifying how many candidates to fetch per channel, and which rewritten or expanded queries to issue.

Given a query plan, the agent uses the retrieval tool, which searches both indexes and merges the results using reciprocal rank fusion (RRF) [29]:

$$\mathrm{RRF}(d) = \sum_{r} \frac{w_r}{k + \mathrm{rank}_r(d)} ,$$

with configurable weights. where:

- d is a candidate dataset record (document);
- r indexes a retrieval channel (e.g., BM25 or dense);
- $\mathrm{rank}_r(d)$ is the rank position of d returned by channel r (lower is better);
- w_r is the weight assigned to channel r;
- k is a smoothing constant controlling the contribution of lower-ranked results.

Depending on the metadata context, it is possible to assign greater or lesser weight to one retrieval channel: the BM25 channel emphasizes lexical overlap and tends to work best when metadata contains distinctive identifiers, codes, or standardized terms, whereas the dense channel emphasizes semantic similarity and is more robust when metadata is short, noisy, or expressed via synonyms and paraphrases [30].

The tool returns the merged candidate list to the agent, which evaluates the candidates based on the similarity score, the variety and diversity of the results, and the genericity of the answer. In this setup, similarity score is the only objective metric. Criteria such as diversity, variety, and genericity are heuristically assessed by the autonomous LLM, without additional evaluation tools. At each iteration, the agent may:

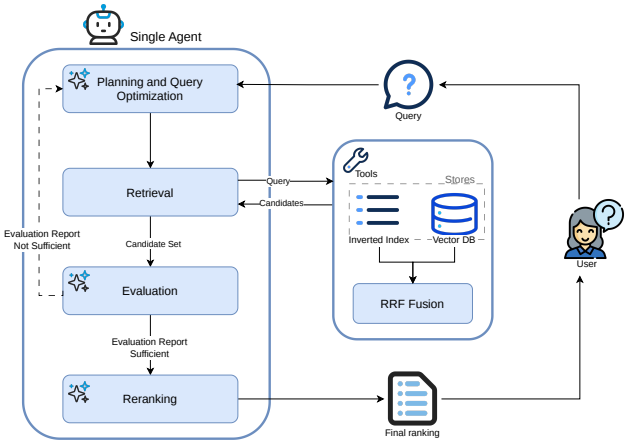

Fig. 1. Single Agent with Plan–Retrieve–Evaluate loop. Dashed arrows denote feedback loops.

- rewrite the query to address vocabulary gaps identified in the current candidates;
- issue additional retrieval calls with refined terms;
- declare the candidate set sufficient.

Upon sufficiency, the agent performs a listwise rerank [6] over the top-N candidates and emits the final ranking. This looping architecture distinguishes it from single-pass RAG pipelines, as the controller adapts its strategy based on intermediate results rather than relying on a single query formulation.

C. Governance and bounded execution

Since the proposed system introduces an explicit control loop, we treat governance as a first-class architectural concern. Each run is bounded by an iteration budget $T_{\max}$, a maximum number of tool calls, and per-channel top-K limits to cap cost and latency, with a deterministic fallback that returns the best candidate set observed so far when budgets are exhausted. For observability and auditability, every iteration emits a structured provenance trace capturing: the agent's query rewrites and plan, retrieval parameters (BM25/dense settings, w_r, k), per-channel ranked lists with scores, the fused RRF list, and the evaluation rationale used to stop or continue. All artifacts are versioned (model, prompts, retriever configuration) to enable replay-based reproducibility and to localize failures to a specific step (planning, retrieval, fusion, or reranking), supporting debugging and controlled evolution of the architecture.

IV. Metadata Augmentation

A core limitation of any retrieval system is the quality of its indexed content. Dataset metadata is authored by data providers with no knowledge of how future users will search, creating a systematic gap between the index and query vocabularies [5], [31]. This limitation is addressed by introducing an offline augmentation step, as shown in Fig. 2. For each dataset record, an LLM generates a set of 5–10 natural-language pseudo-queries that a user may plausibly issue when seeking this dataset.

The augmentation step is proposed as an offline step to improve the perfomances of the single agent architecture: having a more convenient retrieval leads to less iterations needed. This approach draws on the idea of improving table retrieval using question generation [32] and hypothetical document embeddings [33], adapted to leverage LLM generative capacity rather than corpus-based feedback. The rationale follows a shift-left strategy: improving the index offline reduces the number of iterative query rewrites required at query time, thereby improving both latency and result quality during the initial retrieval pass. Pseudo-elements are generated once per dataset record and amortized across all future queries, making the additional LLM cost a fixed offline investment rather than a recurring per-query expense.

The generated pseudo queries can be employed in two different ways that will be evaluated as ablation studies:

- The pseudo-queries are indexed in a dedicated BM25 index and vector store, and are directly used for retrieval instead of the original indexed metadata. Each pseudo-query includes metadata describing the dataset from which it is derived.
- The pseudo-queries are concatenated alongside the original metadata and re-indexed into the BM25 and vector stores.

A. Safeguards

To maintain trustworthiness, the augmentation process is constrained: (i) generated content is derived only from existing metadata (the LLM may not hallucinate external facts); (ii) when metadata is insufficient, the model outputs "unknown" for that facet; (iii) each augmented record stores the model identifier and prompt version for reproducibility; and (iv) a configurable sampling audit manually verifies a random subset of generated pseudo-queries against source metadata.

V. Multi-Agent Decomposition

The single-agent architecture centralizes all reasoning in a single LLM agent. As retrieval tasks become more complex, such as multi-faceted queries requiring geographic and temporal filtering, maintaining and testing a monolithic prompt becomes increasingly challenging. We therefore propose a multi-agent decomposition, shown in Fig. 3, that factors the single agent into four specialized agents.

Recent results also indicate that strong single-agent systems can remain competitive when endowed with capability abstractions (skills/tools) and sufficiently large context windows to preserve coherent state across multi-step execution [34]. In our setting, we hypothesize that as dataset search tasks become more complex (e.g., requiring multi-faceted decomposition, iterative filtering, and tool-intensive reasoning), switching to a multi-agent architecture can yield superior performance by enabling structured collaboration and iterative refinement across specialized roles [18].

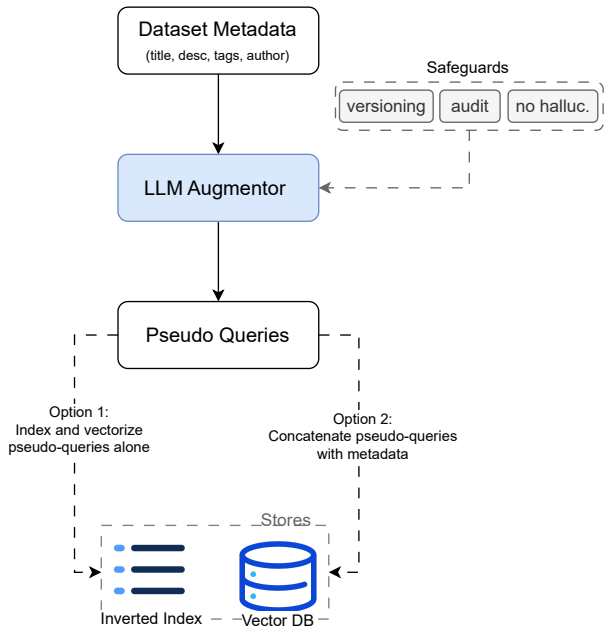

Fig. 2. Offline metadata augmentation pipeline. The LLM Augmentor generates pseudo-queries from dataset metadata, subject to versioning and audit safeguards.

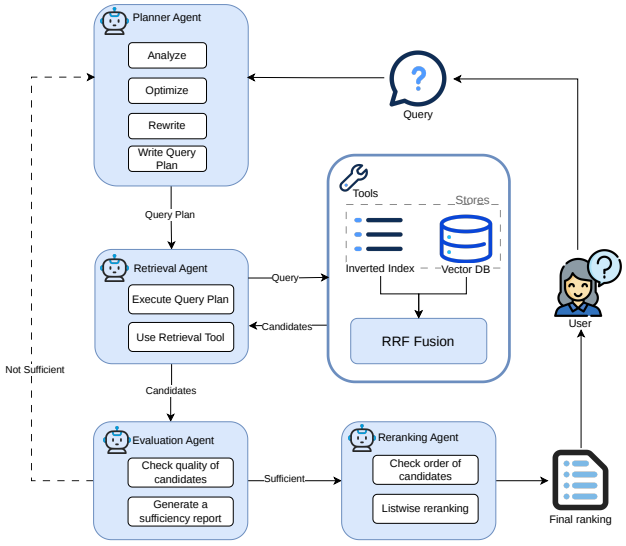

Fig. 3. Multi Specialized-Agent Pipeline. Edge labels are typed inter-agent contracts.

A. Multi-agent workflow

In this case, instead of a single agent switching roles, each role is implemented as a dedicated agent with a well-defined responsibility. The Planner Agent receives the user query, analyzes it, and decides how to optimize it (e.g., by decomposing it into subqueries or adding key terms). The Planner then outputs a query plan which specifies what to search by enumerating the concrete query strings to issue (including expansions and decompositions) and any high-level constraints, but it does not encode low-level retrieval hyperparameters. The Retriever Agent consumes the query plan and autonomously determines how retrieval is executed, including the number of candidates retrieved per channel (lexical vs. dense) and channel weighting. It then executes the resulting retrieval procedure against BM25 and vector stores, producing a candidate set: a list of dataset metadata comprising title, description, tags, and author. The Evaluator Agent scores the candidate set based on the similarity score, the variety and diversity of results, and the genericity of the answer. Then it generates an evaluation report that states whether the candidate set is sufficient and explains the reasons. If not sufficient, the Evaluator triggers the Planner to evaluate again the query in relation to the evaluation report, resulting in a feedback loop. If sufficient, the Evaluator triggers the Reranking agent, which receives the candidate set, checks its ordering and performs a listwise reranking of the set. The final ranking is a ranked list of datasets, each paired with its relative details, and is submitted to the user.

Each inter-agent artifact (query plan , candidate set, evaluation report, final ranking,) is defined by a strict JSON schema. This makes the pipeline composable, independently testable, and observable, as each agent can be swapped, versioned, or mocked without affecting others.

TABLE I
QUALITY-ATTRIBUTE TRADE-OFFS BETWEEN SINGLE-AGENT AND MULTI-AGENT ARCHITECTURES.

Quality attribute	Single-agent	Multi-agent
Modifiability, Testability	M	H
Governance, Observability	M	H
Performance–Cost ratio	H	M
Failure isolation	L	H
Operational complexity	L	H

Note: H = High; M = Moderate; L = Low.

B. Quality-attribute tradeoffs

The multi-agent style enhances modifiability by permitting independent replacement of agents, strengthens governance through per-agent logging and typed contracts for auditability, and improves failure isolation, as a faulty reranker does not compromise the retrieval plan. However, this approach incurs coordination costs, increases the number of LLM calls, and introduces new failure modes, including chain failures and schema drift across agent versions. A summary of the architectural tradeoffs is presented in Table I.

Risks are mitigated via a layered set of governance and robustness tactics. First, strict JSON schema validation can be enforced at each agent boundary, ensuring that malformed outputs and contract drift are detected at the point of production rather than propagating downstream. Second, the multi-agent pipeline benefits from an explicit separation of concerns between planning and execution, which limits cross-agent coupling and clarifies responsibilities. To preserve repeatability and facilitate causal attribution in ablation and regression anal-

TABLE II
EVALUATION VARIANTS: COMPONENT MATRIX.

Component	V1	V2	V3	V4	V5	V6	V7
BM25	P	—	P	P	P	P	P
Dense	—	P	P	P	P	P	P
RRF	—	—	P	P	P	P	P
Planner	—	—	—	I	I	P	P
Retriever	—	—	—	I	I	P	P
Evaluator	—	—	—	I	I	P	P
Reranker	—	—	—	I	I	P	P
Pseudo-queries	—	—	—	—	P	—	P

Note: P = present; I = implemented inside the single-agent controller.

yses, agent autonomy can be bounded by enforcing fixed parameters and logging the runtime-selected parameters. Third, caching can be applied: once a Planner emits a query plan, candidate set, evaluation report and final ranking are cached and keyed by a cryptographic hash of the plan and the index version, thereby reducing redundant LLM calls under repeated executions. Finally, a global iteration budget, both a maximum number of refinement cycles and a wall-clock timeout, bounds cost and precludes non-terminating agent loops.

VI. Evaluation Design

An ablation study is planned out across seven system variants to isolate the contribution of each architectural component, as shown in Table II. Variants V1 and V2 are single-signal baselines, where V1 is based on sparse BM25 and V2 on dense retrieval. V3 combines both signals via Reciprocal Rank Fusion (RRF) to quantify the benefit of hybrid retrieval without any LLM-driven reasoning. V4 introduces the single-agent setup that performs planning, retrieval, evaluation, and reranking. V5 extends V4 with pseudo-query generation to diversify the search space and reduce the number of refinement iterations needed. V6 mirrors V4 but decomposes the single agent into a multi-agent pipeline (planner, retriever, evaluator, reranker) to improve accuracy, observability and failure isolation. Finally, V7 mirrors V5 in the multi-agent setting by adding pseudo-queries on top of the multi-agent pipeline.

A. Retrieval and architectural metrics

The selected metrics are the classic ones for information retrieval systems: Normalized Discounted Cumulative Gain (nDCG) [35], Mean Average Precision (MAP), Recall, and Mean Reciprocal Rank (MRR), evaluated at $k \in \{5, 10, 20\}$.

Beyond retrieval quality, we measure iterations per query, total tool calls, end-to-end latency, token consumption per query, result steadiness across repeated runs (to quantify LLM nondeterminism), and schema validation failure rate (multi-agent only). Stability is operationalized as the Jaccard similarity of top-k result sets across five independent runs with identical inputs, providing a direct measure of the nondeterminism introduced by LLM components.

Architectural qualities are operationalized as follows: modularity via component coupling/cycles and change-impact size (#modules/contracts touched) under controlled refactor tasks; observability via trace/log completeness and time-to-root-cause in injected incident scenarios; failure isolation via blast radius, partial-result availability, and ΔnDCG under fault injection; and governance via provenance completeness (audit replayability) and schema/contract compliance rate (multi-agent only).

B. Large Language Models and Datasets

To carry out the evaluation, prototypes will be developed for both architectures (single- and multi-agent). The prototypes will be based on state-of-the-art (SOTA) models, both open-source and closed-source. For closed-source models, we considered OpenAI GPT 5.2[2], Anthropic Opus 4.6[3], and Google Gemini 3 Deep Think[4]. For open-source models, we considered Kimi K2.5[5] and Alibaba Qwen3.5[6]. In addition, the multi-agent architecture will also be implemented with more economical mid-range models to compare performance and costs with the single-agent system equipped with a SOTA model. Models for this implementation include Anthropic Sonnet 4.6, OpenAI gpt5-mini, and Alibaba Qwen3-coder-next.

The primary benchmark is the ACORDAR 2.0 test collection [27], a large ad hoc dataset-retrieval benchmark with question-style queries and graded relevance judgments over real-world open data portals. We evaluate both its natural-language and keyword queries, and report results on the full corpus as well as a stratified 500-record subset spanning multiple domains. Additional evaluations use NTCIR-15 Data Search [26] and the TARGET table-retrieval benchmark [36] as a stress test of generalization beyond classic ad hoc dataset search.

VII. Conclusion

This work presents a reference architecture for agentic hybrid retrieval in ad hoc dataset search, combining BM25 and dense retrieval under an LLM-orchestrated controller loop. Offline metadata augmentation using pseudo-queries addresses the vocabulary mismatch between user intent and provider-authored metadata without incurring per-query LLM costs. A multi-agent decomposition using typed contracts provides an alternative architectural style when governance, modifiability, and independent testability are emphasized over latency and operational simplicity. A seven-variant ablation framework isolates the contribution of each architectural layer, providing a systematic basis for empirical comparison of single-agent and multi-agent design trade-offs.

References

[1] N. Noy, M. Burgess, and D. Brickley, "Google dataset search: Building a search engine for datasets in an open web ecosystem," in *Proceedings of The Web Conference (WWW)*, 2019, pp. 1365–1375.

[2]https://openai.com/
[3]https://www.anthropic.com/
[4]https://www.google.com/
[5]https://www.kimi.com/
[6]https://qwen.ai

[2] S. Castelo, R. Rampin, A. Santos, A. Freire, and J. Freire, "Auctus: A dataset search engine for data discovery and augmentation," *Proceedings of the VLDB Endowment*, vol. 14, no. 12, pp. 2791–2794, 2021.

[3] A. Chapman and E. Simperl, "Dataset search: A survey," in *Proceedings of the 2019 International Conference on Information and Knowledge Management (CIKM)*, 2019.

[4] N. W. Paton, J. Chen, and Z. Wu, "Dataset discovery and exploration: A survey," *ACM Comput. Surv.*, vol. 56, no. 4, pp. 102:1–102:37, 2024. [Online]. Available: https://doi.org/10.1145/3626521

[5] L.-Y. Gan, A. Das, J. Walker, and E. Simperl, "Keywords are not always the key: A metadata field analysis for natural language search on open data portals," *arXiv*, 2025. [Online]. Available: https://arxiv.org/abs/2509.14457

[6] W. Sun, L. Yan, X. Ma, P. Ren, D. Yin, and Z. Ren, "Is ChatGPT good at search? investigating large language models as re-ranking agents," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023, pp. 14 918–14 937.

[7] Q. Lu, L. Zhu, X. Xu, Z. Xing, S. Harrer, and J. Whittle, "Towards responsible generative AI: A reference architecture for designing foundation model based agents," *arXiv*, 2024. [Online]. Available: https://arxiv.org/abs/2311.13148

[8] M. Hulsebos, W. Lin, S. Shankar, and A. G. Parameswaran, "It took longer than I was expecting: Why is dataset search still so hard?" in *Proceedings of the 2024 Workshop on Human-In-the-Loop Data Analytics (HILDA@SIGMOD)*. ACM, 2024, pp. 1–4. [Online]. Available: https://doi.org/10.1145/3665939.3665959

[9] S. Galhotra, Y. Gong, and R. C. Fernandez, "Metam: Goal-oriented data discovery," in *39th IEEE International Conference on Data Engineering (ICDE)*. IEEE, 2023, pp. 2780–2793. [Online]. Available: https://doi.org/10.1109/ICDE55515.2023.00213

[10] M. Esmailoghli, C. Schnell, R. J. Miller, and Z. Abedjan, "BLEND: A unified data discovery system," in *41st IEEE International Conference on Data Engineering (ICDE)*. IEEE, 2025, pp. 737–750. [Online]. Available: https://doi.org/10.1109/ICDE65448.2025.00061

[11] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," *arXiv*, 2023. [Online]. Available: https://arxiv.org/abs/2312.10997

[12] Y. Yang, P. Carlson, S. He, Y. Qiao, and T. Yang, "Cluster-based partial dense retrieval fused with sparse text retrieval," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24)*, Jul. 2024. [Online]. Available: https://doi.org/10.1145/3626772.3657972

[13] X. Ma, Y. Gong, P. He, H. Zhao, and N. Duan, "Query rewriting in retrieval-augmented large language models," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Dec. 2023, pp. 5303–5315. [Online]. Available: https://aclanthology.org/2023.emnlp-main.322/

[14] Z. Rackauckas, "Rag-fusion: a new take on retrieval-augmented generation," *arXiv*, 2024. [Online]. Available: https://arxiv.org/abs/2402.03367

[15] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-rag: Learning to retrieve, generate, and critique through self-reflection," in *International Conference on Learning Representations (ICLR)*, 2024. [Online]. Available: https://proceedings.iclr.cc/paper_files/paper/2024/file/25f7be9694d7b32d5cc670927b8091e1-Paper-Conference.pdf

[16] Y. Liu, S. K. Lo, Q. Lu, L. Zhu, D. Zhao, X. Xu, S. Harrer, and J. Whittle, "Agent design pattern catalogue: A collection of architectural patterns for foundation model based agents," *Journal of Systems and Software*, vol. 220, p. 112278, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121224003224

[17] Q. Lu, D. Zhao, Y. Liu, H. Zhang, L. Zhu, X. Xu, A. Shi, T. Tan, and R. Kazman, "Agentarceval: An architecture evaluation method for foundation model based agents," *arXiv*, 2025. [Online]. Available: https://arxiv.org/abs/2510.21031

[18] J. Wang, J. Wang, B. Athiwaratkun, C. Zhang, and J. Zou, "Mixture-of-agents enhances large language model capabilities," in *International Conference on Learning Representations (ICLR)*, 2025. [Online]. Available: https://proceedings.iclr.cc/paper_files/paper/2025/file/5434be94e82c54327bb9dcaf7fca52b6-Paper-Conference.pdf

[19] X. Song, H. Wang, Y. Chen *et al.*, "Gradientsys: A multi-agent llm scheduler with react orchestration," *arXiv*, 2025. [Online]. Available: https://arxiv.org/abs/2507.06520

[20] C. L. Wang, T. Singhal, A. Kelkar, and J. Tuo, "MI9 – agent intelligence protocol: Runtime governance for agentic AI systems," *arXiv*, 2025. [Online]. Available: https://arxiv.org/abs/2508.03858

[21] P. Wang, R. Tao, Q. Chen, M. Hu, and L. Qin, "X-WebAgentBench: A multilingual interactive web benchmark for evaluating global agentic system," in *Findings of the Association for Computational Linguistics: ACL 2025*. Vienna, Austria: Association for Computational Linguistics, Jul. 2025, pp. 19 320–19 335. [Online]. Available: https://aclanthology.org/2025.findings-acl.988/

[22] J. Qiu, Z. Liu, Z. Liu, R. Murthy, J. Zhang, H. Chen, S. Wang, M. Zhu, L. Yang, J. Tan, R. Ram, A. Prabhakar, T. Awalgaonkar, Z. Chen, Z. Cen, C. Qian, S. Heinecke, W. Yao, S. Savarese, C. Xiong, and H. Wang, "Locobench-agent: An interactive benchmark for LLM agents in long-context software engineering," *arXiv*, 2025. [Online]. Available: https://arxiv.org/abs/2511.13998

[23] Y. Liu, X. Peng, X. Zhang, W. Liu, J. Yin, J. Cao, and T. Du, "RA-ISF: Learning to answer and understand from retrieval augmentation via iterative self-feedback," in *Findings of the Association for Computational Linguistics: ACL 2024*, 2024, pp. 4730–4749.

[24] T. Yu, S. Zhang, and Y. Feng, "Auto-RAG: Autonomous retrieval-augmented generation for large language models," 2024.

[25] S. Mackie, D. Liu, and S. Culpepper, "Crafting the path: Structured query rewriting for robust information retrieval," arXiv:2407.12529, 2024.

[26] M. P. Kato, H. Ohshima, Y. Liu, and H. Chen, "A test collection for ad-hoc dataset retrieval," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. ACM, 2021, pp. 2450–2456.

[27] M. Risch, N. Reusch, A. Schneiberg, and P. Müller, "ACORDAR 2.0: The largest test collection for ad hoc dataset retrieval," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2024.

[28] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in *International Conference on Learning Representations (ICLR)*, 2023.

[29] G. V. Cormack, C. L. A. Clarke, and S. Büttcher, "Reciprocal rank fusion outperforms condorcet and individual rank learning methods," in *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2009, pp. 758–759.

[30] S. Bruch, S. Gai, and A. Ingber, "An analysis of fusion functions for hybrid retrieval," *ACM Transactions on Information Systems*, vol. 42, no. 1, pp. 1–35, 2023.

[31] H. Zhang, Y. Liu, A. Santos, W.-L. A. Hung, and J. Freire, "Autoddg: Automated dataset description generation using large language models," *arXiv*, 2025. [Online]. Available: https://arxiv.org/abs/2502.01050

[32] H.-P. Liang, C.-W. Chang, and Y.-C. Fan, "Improving table retrieval with question generation from partial tables," in *Proceedings of the 4th Table Representation Learning Workshop*, 2025, pp. 217–228.

[33] W. Gao *et al.*, "Precise zero-shot dense retrieval without relevance labels," arXiv:2212.10496, 2022.

[34] X. Li *et al.*, "When single-agent with skills replace multi-agent systems and when they fail," *arXiv*, 2026. [Online]. Available: https://arxiv.org/abs/2601.04748

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

[36] Y. Zhang *et al.*, "TARGET: A benchmark for table retrieval for generative tasks," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.