Diversed Model Discovery via Structured Table Discovery

Zhengyuan Dong
University of Waterloo
Waterloo, ON, Canada
zhengyuan.dong@uwaterloo.ca

Renée J. Miller
University of Waterloo
Waterloo, ON, Canada
rjmiller@uwaterloo.ca

ABSTRACT

Model cards describe the behavior of models through a mixture of textual descriptions and structured artifacts, including performance, configuration, and dataset tables. Existing model search systems rely predominantly on semantic similarity over text, which can produce homogeneous result sets and limit users' ability to explore alternatives and reason about trade-offs. We argue that model search is inherently comparative: users want models that are aligned at the task level yet differentiated in measurable ways. We hypothesize that this balance requires retrieval over condensed, high-quality evidence rather than verbose descriptions, and much of that evidence is concentrated in structured tables. We present Structured Semantic Search , a table-driven model search framework built on the curated ModelTables benchmark. Given a query, Structured Semantic Search combines semantic model-card retrieval with structure-aware pipelines that discover query-related model-card tables using table discovery operators such as unionability, joinability, and keyword search. Retrieved tables are mapped back to model cards under a controlled top-k budget, enabling fair comparison between text-based and table-based retrieval. Beyond retrieval, Structured Semantic Search adapts table integration to the model-table domain through orientation-aware integration, producing compact integrated views of tables from partially overlapping and sometimes transposed evidence tables. For evaluation, we introduce a nugget-based, auditable protocol that extracts compact evidence items from model cards, matches queries to condition or intent nuggets, and measures evidence coverage and diversity over retrieved model-card candidate sets. This protocol also provides a scalable path toward evidence-based labeling in dynamic model lakes. Experiments on a 597 model-recommendation query set show improved nugget coverage for the structure-aware pipeline compared to semantic baselines.

VLDB Workshop Reference Format:
Zhengyuan Dong and Renée J. Miller. Diversed Model Discovery via Structured Table Discovery. VLDB 2026 Workshop: Tabular Data Analysis (TaDA).

VLDB Workshop Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/RJMillerLab/ModelSearch.

Proceedings of the VLDB Endowment. ISSN 2150-8097.

1 INTRODUCTION

Model search is not document retrieval. Model lakes [45] have emerged as the preferred infrastructure for organizing and sharing machine learning models. Best practice is for each model to have a model card describing training data, evaluation results, and intended usage [39]. Existing model lakes provide some search features, such as HuggingFaceR[4] [36], Modelscope [56], TensorFlow Hub[1], PyTorch Hub[2], DLHub[3]. These systems commonly rely on keyword search, metadata filters, faceted search, or semantic retrieval over model descriptions and model-card text. While these mechanisms may be effective for finding individually relevant models, they provide limited support for constructing comparison-oriented candidate sets of models. However, model search in model lakes often requires more than retrieving individually relevant models [29, 36, 45]. Users may want a set of task-aligned models that also differ in meaningful ways, such as architecture, training corpus, evaluation benchmarks, model variants, or performance trade-offs. This creates a need for diverse model discovery: the result set should remain relevant to the query while exposing non-redundant alternatives for comparison, a topic that has been studied in IR [2], but not model search. This need aligns with the broader information-retrieval view that useful search results should balance relevance with diversity and coverage of user intents

The tension between task alignment and diversity. This observation highlights a fundamental tension. On one hand, retrieved models must be aligned at the task or topic level to remain relevant. On the other hand, users expect diversity in the results, enabling comparison and informed decision-making [63]. Pure semantic similarity optimizes for textual proximity and therefore tends to collapse results around dominant model families (for example, collections of related models developed by an organization) limiting exposure to alternative approaches. This effect is amplified by shared writing templates and reporting conventions: models developed by the same authors or within the same model family often exhibit highly similar narrative descriptions, even when their empirical behaviors differ [11]. This tension suggests that model search should not be optimized for maximal similarity, but for controlled differentiation under task alignment. Achieving this balance requires retrieval signals that go beyond surface-level text similarity and are less sensitive to representational and stylistic bias.

Condensed evidence in model cards. Model cards contain a mixture of narrative text and structured artifacts [39]. While textual descriptions provide contextual information, they are often verbose,

[1] https://www.tensorflow.org/hub
[2] https://pytorch.org/hub/
[3] Deep Learning Hub. https://dlhub.app/

heterogeneous, and shaped by authorial style and templating practices, making direct comparison difficult [4]. In contrast, structured tables, including performance summaries, benchmark results, and configuration listings, concentrate high-density, decision-critical evidence with limited stylistic freedom [25]. These tables encode the core empirical claims of a model and often vary meaningfully even between closely related models [11]. By filtering out irrelevant content and normalizing how evidence is presented, tables provide a more stable basis for comparison. This work explores how such condensed, table-grounded evidence can be leveraged to better support the inherently comparative nature of model search.

Nugget-based evaluation. Model lakes evolve rapidly and user queries vary in specificity: some queries contain explicit conditions (e.g. "4-bit quantized model on X benchmark"), while others are intentionally vague (e.g. "works well on legal documents"). These characteristics make constructing a fixed gold-standard labeling impractical. To evaluate retrieval quality under these constraints, we adopt a nugget-based evaluation [48] with two stages: (1) a card-to-nugget extraction step that pulls compact evidence ("nuggets") from model cards; and (2) a query-to-nugget matching, filtering, and aggregation step that maps queries to condition or intent nuggets and computes a nugget coverage score for candidate sets.

As for nugget definition, prior work varies widely (e.g., sub-questions, atomic facts, or feature-name sets). We adopt the latter and consider a set of fixed features or attributes (specifically, Model, Base model, Model variant, Dataset, Metric name, Metric value). This definition is compatible with a leaderboard-style atomic extraction [21].

Contributions. We summarize our contributions as follows.

- A table-driven model discovery framework that complements semantic model-card retrieval by searching and integrating structured tables extracted from model cards, including both model-card anchoring (*NL2Card2Tab2Card*) and direct table anchoring (*NL2Tab2Card*).
- A nugget-based evaluation metric and two-stage pipeline (nugget extraction and prompt-assisted query-to-nugget matching) that measures evidence coverage and diversity; the metric is explicitly scoped to evaluate the nuggets extracted from retrieved candidate sets (not full model-card processing) and supports evidence-based labeling in dynamic model lakes.
- A practical integration strategy that is orientation-aware (handling tables that have been transposed) to improve comparability across retrieved evidence; from a downstream-integration perspective, the retrieved set should be visibly relevant yet diverse, and integration provides a convenient, user-facing view for side-by-side comparison.
- An end-to-end implementation that allows inspection of retrieved tables and integration views, together with an adapted model-recommendation query set derived from paper-recommendation data; experiments using this query set show improved nugget coverage for our pipeline compared to semantic baselines.

[4] https://huggingface.co/templates/model-card-example

Our work is evaluated over 60K models from HuggingFace [11] and the system will be demonstrated at the workshop.

2 RELATED WORK

2.1 Model Lake

Model lakes are large collections of heterogeneous machine learning models and their associated artifacts, as envisioned by Pal et al. [45]. The model-lake literature spans tasks such as model attribution and provenance tracking [37, 41, 58], model versioning and lineage analysis [28, 54], model search and retrieval [30, 35], benchmarking and reporting [31, 39], and documentation generation [34]. Model cards are a central source of model evidence: they record model details, intended use, training data, evaluation results, and limitations [39]. Yet studies show that such documentation is often incomplete, inconsistent, or hard to compare across models [31], which is why prior work has explored metadata representations for queryable repositories [29], task and model embeddings for retrieval [1], content-based model search [35], graph-based model selection [30], and LLM-based orchestration over model descriptions [53]. More recent work extends the selection side of this space by ranking unseen models on unseen datasets from leaderboard-style tuples [4]. Together, these works frame model search as a structured model-selection problem driven by heterogeneous documentation rather than text similarity alone.

2.2 Data Discovery

Data discovery studies how users find useful datasets and tables in large, heterogeneous data lakes [14, 16, 42]. Table annotation research treats table labeling and schema-level description as a core step for making heterogeneous tables searchable [26]. For tabular data, table search aims to retrieve tables relevant to a query [6, 7, 27], while joinable search aims to identify tables that can be linked through shared entities or values [9, 62]. Unionable table search focuses on tables with compatible schemas or semantically aligned columns [15, 18, 23, 43]. Unified discovery systems combine these operators in a single system [13], and table integration completes the pipeline by aligning and combining related tables into consolidated views for downstream analysis and comparison [24]. Complementary benchmarking work evaluates tabular encoders for downstream discovery tasks such as joinable and unionable search [46]. This body of work motivates treating table search and table integration as complementary parts of one discovery pipeline when the goal is to assemble comparable evidence from fragmented tabular sources.

2.3 Nugget Analysis and Evaluation

Traditional retrieval metrics such as nDCG [19], MAP [52], and RBP [40] evaluate relevance at the document level, but they do not directly measure whether a retrieved set covers the full breadth of a user's information need. Nugget-based evaluation addresses this limitation by decomposing answers into atomic information units in QA [33, 57], while the pyramid method evaluates summarization outputs through summary content units [44]. In retrieval, this coverage perspective is closely related to search result diversification, where α-nDCG measures novelty and redundancy-aware gain [8],

IA-ERR models intent-aware ranking quality [5], and Subtopic Recall measures how many distinct subtopics are covered [61]. Recent RAG and report-generation evaluations further adopt nugget-based coverage, since missing evidence in retrieval can lead to incomplete generated answers [47, 51]. This coverage perspective is also relevant to model search, where effective comparison requires not only retrieving relevant model documentation, but also covering complementary evidence about capabilities, benchmarks, datasets, metrics, and constraints.

2.4 Leaderboard Generation

Leaderboards are widely used to summarize experimental progress by organizing methods, datasets, metrics, and performance results into comparable rankings. Prior work extracts tasks, datasets, evaluation metrics, and numeric scores from machine learning papers [17, 21], and follow-up work extends this with table-centric extraction and organization [20, 60]. More recent work studies LLM-based performance tracking and benchmark construction for scientific leaderboards [50, 55, 59]. We borrow the tuple-oriented view from this literature: it is a convenient way to represent performance evidence, but leaderboard construction itself is not the target of this work.

3 METHODOLOGY

Current deployed model search for keyword or natural language queries uses model cards [5]. We will use this as our baseline (*NL2Card*) that we call Unstructured Semantic Search. We propose two table-aware candidate-generation pipelines, *NL2Card2Tab2Card* and *NL2Tab2Card*, that we call Structured Semantic Search. We describe each below.

3.1 Unstructured Semantic Search

NL2Card can be done using basic semantic search over the semi-structured model cards of a model lake. In Figure 1 on the left, we depict this traditional semantic search for model cards (and their associated models) in Pipeline 1.

Our experiments will use three implementations of semantic search: dense, sparse, and hybrid. Dense retrieval is implemented with a Sentence-BERT encoder [49] and FAISS [12]. Sparse retrieval is implemented with Pyserini [32] and the hybrid variant retrieves an expanded sparse candidate pool before dense reranking. The experiments report results on all three variants.

3.2 Model-Card Anchoring: NL2Card2Tab2Card

To improve the quality and diversity of model search, NL2Card2Tab2Card leverages the knowledge-rich tables found in a model lake. We first use *NL2Card* (semantic search) to find an anchor model card, that is, the top-1 *NL2Card* ranked card. We then use the tables associated with the anchor card in a table discovery search process described formally below. These tables are associated with one or more models. Our pipeline, called Structured Semantic Search, uses a query-to-card-to-table-to-card workflow and is shown in Figure 1 Pipeline 2. We detail each step below.

[5] https://huggingface.co/; https://huggingface.co/spaces/librarian-bots/huggingface-semantic-search

This design isolates the effect of table discovery: the semantic retrieval of an anchor model card ensures that we are finding a model associated with query task, while table discovery expands the candidate set of models through structured evidence such as shared benchmarks, metrics, identifiers, and configuration attributes.

3.2.1 Structure-Aware Table Discovery. NL2Card2Tab2Card begins with an anchor model card selected by Unstructured Semantic Search. Because table discovery requires structured evidence, the anchor step is constrained to model cards with at least one associated table. For each anchor table, defined as a table associated with an anchor card, NL2Card2Tab2Card searches the model table lake using table discovery operators implemented in Blend [13]. Specifically, we use three Blend operators for keyword search over tables (data and metadata), joinable table search, and unionable table search.

Keyword Table Search. Keyword search retrieves tables containing tokens from a query set. In model tables, semantic labels and identifiers are typically concentrated in headers and first-column values (examples include benchmark task names, model, or dataset identifiers), while interior cells often contain numeric measurements and scalar values; both are informative for discovery, but play different roles. We therefore construct keyword queries over the header and first column of an anchor table, execute Blend's value-based table keyword search operator, and rank candidate tables by matched-token frequency.

Joinable Table Search. Joinable table search retrieves tables that can be joined with a column of an anchor table [62]. In model tables, joinable columns often correspond to model names, dataset names, task names, or benchmark identifiers. We use the first column of an anchor table as the query column and retrieve joinable tables using Blend. Tables with larger overlap in the join columns are ranked higher.

Unionable Table Search. Unionable table search retrieves tables whose columns can be aligned with an anchor table so that their contents can be meaningfully unioned (or outer-unioned if some columns do not align) [43]. As an example, this operator is especially useful for finding benchmark or configuration tables that report comparable attributes for different models. We rank candidate tables by the number of distinct anchor columns that can be aligned.

3.2.2 Mapping Tables Back to Model Cards. Table discovery naturally returns tables, but the final retrieval task needs to return model cards. After table discovery, we have a ranked list of tables each of which is associated with one or more model cards. A table can be associated with more than one card if, for example, it is from a paper referenced by two or more model cards [11]. For each table, we select a single model card. To do this, we select the model card with the highest semantic retrieval similarity (using Unstructured Semantic Search) to the query. This table-wise top-1 selection ensures that each retrieved table contributes a single representative card, which avoids inflating the candidate set with multiple cards supported by the same table evidence. The resulting model-card candidates (one per table) are then also ranked by their semantic query similarity and the top-k selected.

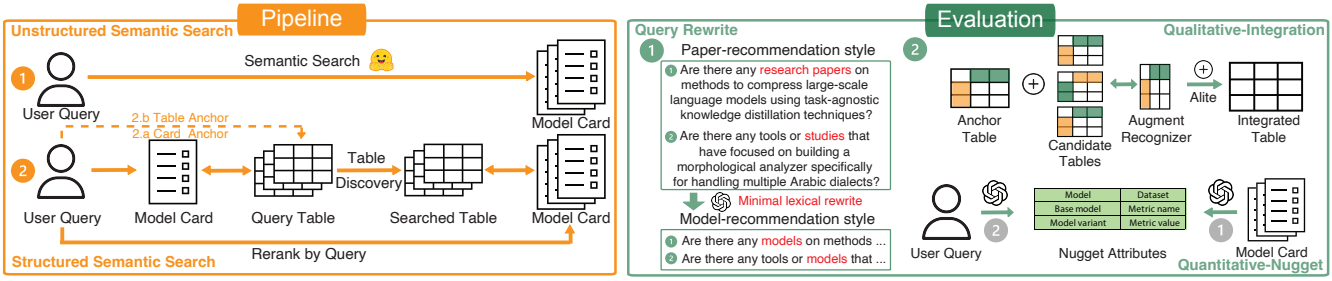

Figure 1: Overview of our table-driven model search and evaluation workflow. The pipeline compares text-only model-card retrieval (*NL2Card*) with table-mediated retrieval through either a model-card anchor (*NL2Card2Tab2Card*) or a direct table anchor (*NL2Tab2Card*). Evaluation uses rewritten paper-style queries and measures candidate quality through table integration and nugget coverage.

3.3 Direct Table Anchoring: NL2Tab2Card

NL2Card2Tab2Card uses semantic model-card retrieval to select an anchor model card before table discovery. We additionally evaluate a direct table-anchor variant, *NL2Tab2Card*, to test whether the table-mediated retrieval idea depends on first passing through a model-card anchor. Given a natural-language query, NL2Tab2Card first retrieves an anchor table directly from the model-table lake. After the anchor table is selected, the downstream pipeline is identical to NL2Card2Tab2Card: the anchor table is used as the query table for keyword, joinable, or unionable table discovery; retrieved tables are mapped back to model cards; and the resulting model-card candidates are reranked under the same top-k budget. The end-to-end algorithms are provided in the appendix of the full version [10].

Keyword-based table anchoring. The keyword-based NL2Tab2Card variant treats the natural-language query as a keyword set and applies table keyword search over the indexed model-table lake. The top-ranked table is used as the anchor table. This variant tests whether lexical overlap between the query and table values or metadata is sufficient to initialize table-mediated expansion.

Embedding-based table anchoring. The embedding-based NL2Tab2Card variant embeds each table into the same dense vector space used for semantic retrieval. At inference time, the query is encoded with the same encoder, and the nearest table by dense similarity is selected as the anchor table. This variant tests whether dense table representations can provide a stronger bridge between natural-language information needs and structured table evidence.

4 MODEL RANKING EVALUATION STRATEGY

Our goal is to compare the evidence surfaced by the baseline, Unstructured Semantic Search and our new table-based search, Structured Semantic Search. We will not consider how to achieve a static model-ranking benchmark (which to the best of our knowledge does not exist). This matters because model lakes are continuously expanding: any fixed ground-truth annotation quickly becomes stale as new models are added. We therefore need a comparative evaluation method that is query-aware, evidence-oriented, and stable under growth of the model lake. We present a quantitative comparative evaluation strategy in Section 4.1 followed by a table-based case studies proposal in Section 4.2.

4.1 Nugget-based Quantitative Evaluation

We adopt a recent nugget-based strategy from information retrieval [47]. The nugget formulation lets us represent query-relevant evidence as compact, auditable units rather than as coarse document labels. While this strategy has recently been proposed for documents, to the best of our knowledge, it has not been used for model cards. In our setting, the same query may be satisfied by several model cards that differ only in fine-grained evidence, so the evaluation will count the evidence units explicitly. The `Evaluation` block on the right side of Figure 1 illustrates the nugget-based evaluation setting. Before giving the formal details, we present an example.

Example 4.1. Consider the query: "Could you recommend models that evaluate the performance decline in various language models, like BLOOM, under 4-bit integer columnar weight-only quantization?" This and similar model queries refer to standard concepts like model variant ("quantization" is a model variant) or metrics (in this query, the metric name is "quantization bits" and its value is "4-bit"). We define several common concepts found in model searches and model cards. These form the nuggets that we will extract from the model cards. We also map the query to nuggets viewing the query as a set of constraints that the nuggets of a retrieved model card should satisfy. For this query, over the HuggingFace model lake we will use in our experiments (Section 5.1), the dense retrieval version of Unstructured Semantic Search returns 10 candidate model cards, including **inventbot/Mixtral-8x7B-Instruct-v0.1-offloading-demo**, Card-level nugget extraction yields 52 raw nugget rows in total. For example, from the model, we extract a nugget stating the model variant is quantization, the metric quantization value is 4-bit, the metric groupsize has value 64, the metric compression has value 0, and many others. The first nugget matches the query's nugget. This model matches the query need.

Nugget Definition. The nugget concept is designed to correspond to atomic evidence units that are used to represent fine-grained information needs. We adapt that idea to model search by defining each nugget as a structured tuple with a fixed set of six attributes: `model`, `base model`, `model variant`, `dataset`, `metric name`, and `metric value`. A nugget then is a 6-tuple with one value

(or null) for each of the six attributes. Notice some values can be null as not all models have base models and not all model cards mention the model variant. For a model card c, we denote by $\mathcal{N}(c)$ the set of nuggets extracted from c.

This fixed attribute list is important for three reasons. First, it standardizes the evidence representation so that different model cards can be compared in the same schema. Second, it is faithful to the structure of model cards, where these fields are commonly used to describe performance and model identity [39]. Third, it makes the evaluation transparent: the same query always maps to the same kind of evidence, which is easier to inspect and trust than an open-ended free-text annotation [51].

Example 4.2. Consider the model card for **luisra/Kimi-K2-Instruct-4bit**, two example nuggets extracted from this card are: (*luisra/Kimi-K2-Instruct-4bit*, *moonshotai/Kimi-K2-Instruct*, *null*, *null*, *null*, *null*) and (*luisra/Kimi-K2-Instruct-4bit*, *null*, *null*, *LiveCodeBench v6*, *Pass@1*, *0.537*). The first nugget captures the model lineage by linking the card to its base model, while the second nugget records benchmark performance on LiveCodeBench v6. Together, these nuggets illustrate how a single model card can contain multiple kinds of structured evidence under the same fixed schema.

Nugget Extraction. Given a model card, we extract nuggets by feeding the full card content into a prompt-based extractor that is instructed to populate our fixed nugget schema with six attributes. The card may contain performance evidence in tables, tags, benchmark summaries, or evaluation subsections, so the prompt is designed to recover structured evidence from heterogeneous formatting. We then normalize each extracted item into an instantiated nugget under the fixed schema.

The output of this stage is a nugget table containing a seventh attribute storing the model card identifier. Because the schema is fixed and the prompt operates on the full model card, extraction only needs to be run for newly added model cards at ingestion when the model lake expands; existing nuggets do not need to be recomputed. This makes the representation suitable for continuously growing model lakes. The full prompts used for nugget extraction, query-to-nugget mapping, and nugget filtering are provided in the appendix of the full version [10].

Query-to-Nugget Mapping. The query is not itself a nugget, so we introduce an intermediate mapping from query text to nugget constraints. The mapping identifies a subset of the nugget attributes that are relevant to the query and, when the query is specific enough, additional constraints on the attribute value of the nugget. For vague queries, the mapping is broad and only requires attribute compatibility. For detailed queries, the mapping includes both attributes and values, such as a benchmark name, a quantization level, or a dataset name.

We use a prompt-based method to map a query q to a standardized representation $\phi(q)$ that contains the relevant nugget attributes and any associated constraints. The query-relevant nugget set is then the subset of instantiated nuggets that match $\phi(q)$ after normalization and disambiguation. In principle, this could be expressed as a SQL-style filter over the nugget table, but we do not rely on a literal exact matching in practice. Many model search queries are vague or semantically under-specified, and exact field equality would miss valid evidence. We therefore use a prompt-based filter that interprets the query intent and normalizes semantically equivalent mentions before selecting the query-relevant nuggets. The prompt input, model output, and post-processed representation are recorded for each query so the mapping is auditable and reproducible.

Candidate-Set Scoring. We use a single quantity-based score to compare retrieval methods at the model card level. For a retrieved candidate set of model cards $R_q = \{m_1, \ldots, m_k\}$, we count the number of unique query-relevant nuggets covered by at least one retrieved model card. To do this, we first take the set union of the nuggets of all model cards in the candidate set.[6] The *Nugget Score* $\text{Score}(R_q, q)$ is then the number of nuggets in this set satisfying the query constraints $\phi(q)$.

This score treats overlap carefully. If the same nugget appears in multiple retrieved model cards, it is counted only once at the set level, preventing redundant evidence from inflating the result. This is especially important for similarity-based retrieved sets, where highly similar model cards often contain overlapping evidence. The score therefore directly measures how much distinct query-relevant evidence the candidate set surfaces. We use this lightweight coverage score instead of document-level ranking metrics such as nDCG, MAP, or RBP because our setting does not have complete relevance judgments or complete nugget extraction over the entire model lake. Exhaustively extracting and validating nuggets for all model cards would be expensive, especially as model lakes evolve. Our goal is therefore comparative: given the outputs of different retrieval methods under the same top-k budget, we measure how much query-relevant evidence each output set contains. By filtering to query-relevant nuggets rather than counting all extracted nuggets, the score emphasizes evidence that directly supports the query instead of simply rewarding longer or more heavily documented model cards.

In our experiments, we compute this score for the Unstructured Semantic Search methods and for the Structured Semantic Search methods and compare their nugget counts under the same top-k budget. Because the score is based on set coverage rather than document rank, it reflects the amount of evidence available for downstream inspection by a model lake user.

4.2 Table-based Case Studies

In addition to the nugget-based model-card score, we integrate the retrieved tables and present them to the user as case studies. The table integration block in Figure 1 illustrates the table-alignment view used for manual inspection. Retrieved tables are often partially overlapping, noisy, or transposed due to augmentation, so the integration step must be orientation-aware and iterative. Our implementation is built on ALITE [22], a scalable approach to integrating data lake tables that maximally integrates facts scattered across tuples in different tables.

Table orientation. To handle tables that may be transposed, we add an orientation-recognition step before integration. For each table pair, we compare header keywords in one table against both

[6] This is the 6-tuple nuggets without the model card id.

the header row and the first column of the other table. When the overlap pattern suggests that the two tables are semantically aligned but transposed, we transpose one table before integration. This patch prevents direct integration from producing two poorly aligned blocks with many missing values. The orientation-aware integration algorithm is summarized in the appendix of the full version [10].

For the table-driven pipeline, integration is well matched to the retrieval process because the candidate tables have already been selected through table discovery operators that encode table-level relatedness. Unionable search tends to return schema-compatible tables, while joinable and keyword search expose value or token overlap that can still support partial alignment. By contrast, tables collected only after model-card retrieval may be less compatible; in these cases, the integrated view is still useful because unmatched or sparse regions appear as null blocks, making incompatibility visible rather than hiding it in long model-card text.

The resulting integrated view is the table-level counterpart to the nugget-based candidate-set score: one asks how much distinct evidence is surfaced, and the other asks whether that evidence can be organized into a coherent, comparable table.

5 EXPERIMENTS

We evaluate our proposed structure-aware pipelines against text-only model-card retrieval on a set of almost 600 model-search queries. The experiments compare direct model-card retrieval (*NL2Card*), model-card-anchored table expansion (*NL2Card2Tab2Card*), and direct table-anchored expansion (*NL2Tab2Card*). We first present the model lake we use and the queries. We then present our quantitative evaluation, which measures the number of nuggets satisfying a query returned by each search method (Section 5.2). We conclude this section with two examples to better illustrate the search.

5.1 Dataset

Model Lake. We use the curated ModelTables corpus [11] as the model lake. The corpus includes over 60K model cards extracted from HuggingFace and provides a high-quality, deduplicated set of model tables associated to these model cards. The tables were extracted from the model cards, from code repositories referred to in the model cards, and from papers referenced in the model card. The deduplication is important for table search: it prevents the retrieval stage from repeatedly surfacing identical tables and therefore improves the diversity and utility of the evidence returned by table discovery. We preprocess the tables keeping only compact tables with fewer than 200 rows and 100 columns, since the smaller tables are more likely to summarize model behavior, benchmark results, configuration attributes, or deployment constraints. Because the same table may still be associated with multiple model cards, we keep the full table-to-card linkage during retrieval and mapping so that reranking can select one representative model card per retrieved table. This makes the final candidate set for Structured Semantic Search more diverse than Unstructured Semantic Search, since it is driven by distinct table evidence rather than repeated copies of the same table.

Query corpus. We derive our model-search queries from LitSearch [3], a scientific literature retrieval benchmark whose 597 queries are phrased as paper-recommendation requests. To adapt these queries to model search, we apply a prompt-based rewrite that makes the smallest natural lexical edit while preserving the original information need. The rewrite prompt preferentially substitutes paper-oriented terms such as `paper`, `studies`, `publications`, `articles`, and `literature` with model-oriented terms such as `model`, `method`, `approach`, `benchmark`, or `task` when appropriate, but otherwise keeps the wording and structure unchanged. This yields a query set that retains the exploratory and comparative character of LitSearch while shifting the target from papers to models. Importantly, we do not introduce new constraints during rewriting, so the adapted queries remain close to the original retrieval intent instead of becoming synthetic model-search prompts. The Query Rewrite block in Figure 1 shows how we construct the evaluation query corpus.

We additionally annotate the adapted queries with a non-factoid query taxonomy for descriptive analysis; this analysis is orthogonal to the retrieval pipeline. Details on this and all prompts are provided in the supplementary material [10].

5.2 ModelCard-Level Quantitative Evaluation

For each adapted query, we run all retrieval methods and collect the returned top-k model-card candidates. The methods include three *NL2Card* variants (dense, sparse, and hybrid), three *NL2Card2Tab2Card* variants using keyword, joinable, and unionable table discovery, and *NL2Tab2Card* variants that select anchor tables directly from the query using dense table-embedding search before applying the same downstream table discovery operators. We then compute the nugget-count score defined in Section 4 for each returned set. Because the same query can favor different evidence patterns, we report both the per-query distribution and the aggregate summary. At the per-query level, the same nugget is counted only once even if it appears in multiple retrieved cards, so the score reflects distinct evidence rather than redundant copies of the same fact. At the aggregate level, the mean coverage tells us which methods surface the broadest evidence mass across the full benchmark.

Note that even for top-1 queries that return a single model card, there may possibly be hundreds of unique nuggets satisfying a query. As one example, for the query `"Could you suggest models that investigate how many evidence sentences are needed for document-level RE?"`, the nugget creation recognizes that datasets are required to answer this query. Furthermore, the model `fblgit/juanako-7b-v1` has been run on dozens of benchmark datasets and reports several performance metrics on each leading to 134 nuggets being created that match the query.

To understand how retrieval strategy changes the models returned, we show two complementary views of the results in Figure 2. The top (blue) panel is method-centered: it aggregates the number of nuggets satisfying the query over all adapted queries and shows the number of distinct nuggets each retrieval method tends to surface, independent of any single query. The bottom panel is query-centered: for each query, we compare all methods and count how often each one lands at each rank under the same

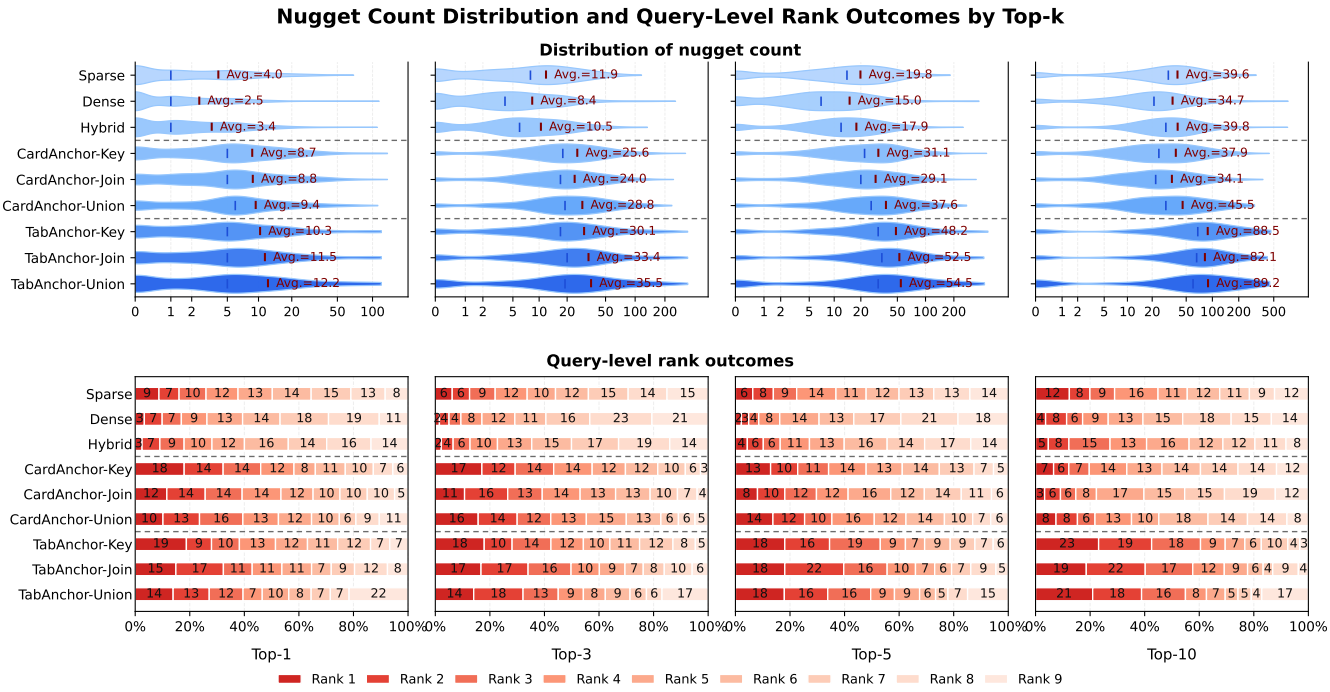

Figure 2: Nugget coverage and query-level rank-share outcomes for text-only model-card retrieval (*NL2Card*), model-card-anchored table expansion (*NL2Card2Tab2Card*), and direct table-anchored expansion (*NL2Tab2Card*) under top-1, top-3, top-5, and top-10 budgets. The top panels show per-query nugget-count distributions; blue ticks mark medians and red ticks mark means. The bottom panels show, for each method, the fraction of queries at each within-query rank. Direct table anchoring consistently improves nugget coverage, with the unionable TabAnchor variant producing the highest mean coverage across all budgets.

top-k budget. This split matters because the structure-aware family is not uniform. Unionable, joinable, and keyword search impose different structural constraints, so their behavior depends on how directly the query aligns with the available table schemas and value patterns. Likewise, the Unstructured Semantic Search family is not interchangeable: sparse retrieval often remains competitive because exact lexical overlap can still capture strong task alignment in this benchmark.

The top panel illustrates which methods surface the most nugget evidence on average, and the bottom panel illustrates which methods are most often near the top on a per-query basis. That second question is important because a method can have a strong average without being the most consistent winner, and vice versa. Across all retrieval budgets, direct table anchoring yields the strongest nugget coverage. The improvement is visible at top-1 and becomes larger as the budget increases: the unionable TabAnchor variant reaches 54.5 nuggets at top-5 and 89.2 nuggets at top-10, substantially above both direct model-card retrieval and model-card-anchored table expansion. This suggests that the first anchor is a major bottleneck: directly locating a relevant table from the query places the downstream table expansion step in a more evidence-rich region of the table lake.

Anchor selection ablation. The NL2Tab2Card results isolate the role of the anchor selection step. Compared with NL2Card2Tab2Card, which reaches table discovery through an anchor model card, NL2Tab2Card directly anchors the query in the table lake. We instantiate this direct anchor selection by embedding each table and selecting the nearest table to the query, then holding the downstream table discovery operator fixed across keyword, joinable, and unionable variants. This direct table anchoring is especially strong when combined with unionable table discovery, suggesting that table-level representations can serve as an effective bridge between natural-language queries and model-card evidence. The result strengthens the main hypothesis of this paper: the benefit comes not only from retrieving semantically similar model cards, but from using structured table evidence as a retrieval medium for discovering comparison-ready and diverse model candidates.

5.3 Table-Level Case Studies

We use a resource-constrained model selection query, "`Which image classification models are lightweight enough for deployment on edge devices?`", as a representative example in Figure 3. The main insight is that the Unstructured Semantic Search methods are generally returning less diverse sets of models but also return models whose cards may not contain interesting structured evidence useful to a data scientist in comparing the models. Even when relevant cards are retrieved, users may still need to read long textual descriptions, and the information is often presented in inconsistent formats across cards, which makes quick comparison difficult. In contrast, using Structured Semantic

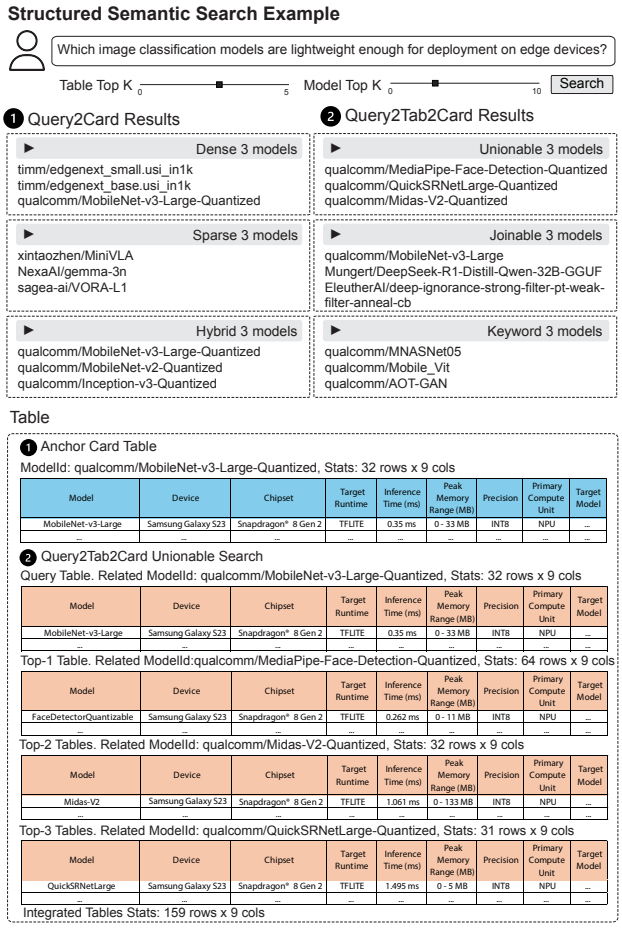

Figure 3: (1) Unstructured Semantic Search returns less diverse result sets of models than (2) Structured Semantic Search. In addition, Structured Semantic Search expands a query-aligned seed table (from the anchor card) with related tables (we show the results of using unionability as the relatedness measure) from related models, enabling a broader and more transparent comparison across models. The integration (union) of these tables also provides the user with task-relevant information on the performance of related models augmenting the card-search with valuable structured information.

Search methods, we ensure the retrieved evidence is table-backed and thus already organized in a more consumable way. Furthermore, we present (and integrate) the tables used in our search as illustrated in Figure 3. In this example, this enables the system to produce a broad yet still coherent comparison view over attributes such as device, chipset, runtime, latency, memory usage, precision, and compute unit. This is analogous to benchmark curation platforms such as Papers with Code[7], and suggests that table-centric retrieval can support the automatic construction of benchmark-style comparison tables by integrating compatible evidence from different model-card tables.

[7] https://paperswithcode.com

6 CONCLUSION

We presented Structured Semantic Search, a table-centric framework for model search in model lakes. Rather than relying only on model-card-level semantic retrieval, Structured Semantic Search treats tables as searchable and integrable evidence units, which makes it possible to retrieve structured model information, surface more diverse evidence (and models), and assemble comparison-ready candidate sets.Our evaluation uses 597 published LitSearch paper recommendation queries adapted to model recommendation, and our nugget-based quantitative evaluation measures how much query-relevant evidence each retrieval method surfaces at the model-card level. A small set of table-level case studies complements the quantitative results by showing how the structure-aware pipeline produces coherent, comparison-ready integrated views. Taken together, the nugget-based quantitative results and the table-level case studies support the central claim that table-centric retrieval is a useful complement to information-retrieval-based semantic model search for evidence-grounded decision making.

There are several directions for future work. First, future work should consider richer anchor selection and query-aware operator selection. This paper evaluates both model-card anchoring and direct table anchoring, but future systems could adaptively choose between them based on query intent and the available table evidence. Similarly, our experiments run unionable, joinable, and keyword search as separate retrieval variants; a deployed system could select or combine these operators based on query intent and the available table evidence. Second, to integrate tables, we have used in this paper Alite [22], a scalable approach to integrating data lake tables that maximally integrates facts scattered across tuples in different tables. However, Alite does not work well if one table is the transpose of another (or more generally if they exhibit schematic heterogeneity where data in one table is used as headers in another) [38]. Such heterogeneity is common in model lakes [11] and more research is needed on how to best integrate these tables. Third, many model cards remain incomplete or only partially table-backed, so future systems should infer or augment missing structure to improve search and integration over incomplete cards. Fourth, the majority of our queries are evidence-based , as reported in the appendix of the full version [10], and it would be interesting to understand whether the search performance trade-offs change for workloads with different query intents. Finally, our nugget-based score provides a lightweight way to compare retrieval strategies under the same top-k budget without requiring full-lake relevance annotation. By focusing on query-relevant and deduplicated evidence, it captures the distinct evidence surfaced by each method while remaining practical for evolving model lakes. Future work can extend nugget extraction more broadly to support benchmark-style precision and recall analysis.

ACKNOWLEDGMENTS

We acknowledge the support of the Canada Excellence Research Chairs (CERC) program. We thank the anonymous reviewers for their insightful comments and suggestions.

REFERENCES

[1] Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless Fowlkes, Stefano Soatto, and Pietro Perona. 2019. Task2Vec: Task Embedding for Meta-Learning. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 6429–6438. https://doi.org/10.1109/ICCV.2019.00653

[2] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. 2009. Diversifying search results. In *Proceedings of the second ACM international conference on web search and data mining*. 5–14.

[3] Anirudh Ajith, Mengzhou Xia, Alexis Chevalier, Tanya Goyal, Danqi Chen, and Tianyu Gao. 2024. Litsearch: A retrieval benchmark for scientific literature search. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. 15068–15083.

[4] Rui Cai, Weijie Jacky Mo, Xiaofei Wen, Qiyao Ma, Wenhui Zhu, Xiwen Chen, Muhao Chen, and Zhe Zhao. 2026. ModelLens: Finding the Best for Your Task from Myriads of Models. *arXiv preprint arXiv:2605.07075* (2026).

[5] Olivier Chapelle, Shihao Ji, Ciya Liao, Emre Velipasaoglu, Larry Lai, and Su-Lin Wu. 2011. Intent-based diversification of web search results: metrics and algorithms. *Information Retrieval* 14, 6 (2011), 572–592.

[6] Martin Pekár Christensen, Aristotelis Leventidis, Matteo Lissandrini, Laura Di Rocco, Renée J. Miller, and Katja Hose. 2025. Fantastic Tables and Where to Find Them: Table Search in Semantic Data Lakes. In *Proceedings 28th International Conference on Extending Database Technology, EDBT 2025, Barcelona, Spain, March 25-28, 2025*, Alkis Simitsis, Bettina Kemme, Anna Queralt, Oscar Romero, and Petar Jovanovic (Eds.). OpenProceedings.org, 397–410. https://doi.org/10.48786/EDBT.2025.32

[7] Christina Christodoulakis, Eric B. Munson, Moshe Gabel, Angela Demke Brown, and Renée J. Miller. 2020. Pytheas: Pattern-based Table Discovery in CSV Files. *Proc. VLDB Endow.* 13, 11 (2020), 2075–2089. http://www.vldb.org/pvldb/vol13/p2075-christodoulakis.pdf

[8] Charles LA Clarke, Maheedhar Kolla, Gordon V Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. 2008. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. 659–666.

[9] Yuyang Dong, Chuan Xiao, Takuma Nozawa, Masafumi Enomoto, and Masafumi Oyamada. 2023. DeepJoin: Joinable Table Discovery with Pre-trained Language Models. *Proc. VLDB Endow.* 16, 10 (2023), 2458 – 2470. https://doi.org/10.14778/3603581.3603587

[10] Zhengyuan Dong and Renée J. Miller. 2026. Diversed Model Discovery via Structured Table Discovery: Full Version. https://github.com/RJMillerLab/ModelSearch/blob/main/papers/full_version.pdf. Accessed: 2026-07-07.

[11] Zhengyuan Dong, Victor Zhong, and Renée J. Miller. 2025. ModelTables: A Corpus of Tables about Models. *CoRR* abs/2512.16106 (2025). https://doi.org/10.48550/ARXIV.2512.16106 arXiv:2512.16106

[12] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2025. The faiss library. *IEEE Transactions on Big Data* (2025).

[13] Mahdi Esmailoghli, Christoph Schnell, Renée J. Miller, and Ziawasch Abedjan. 2025. BLEND: A Unified Data Discovery System. In *41st IEEE International Conference on Data Engineering, ICDE 2025, Hong Kong, May 19-23, 2025*. IEEE, 737–750. https://doi.org/10.1109/ICDE65448.2025.00061

[14] Grace Fan, Jin Wang, Yuliang Li, and Renée J. Miller. 2023. Table Discovery in Data Lakes: State-of-the-art and Future Directions. In *Companion of the 2023 International Conference on Management of Data, SIGMOD/PODS 2023, Seattle, WA, USA, June 18-23, 2023*, Sudipto Das, Ippokratis Pandis, K. Selçuk Candan, and Sihem Amer-Yahia (Eds.). ACM, 69–75. https://doi.org/10.1145/3555041.3589409

[15] Grace Fan, Jin Wang, Yuliang Li, Dan Zhang, and Renée J. Miller. 2023. Semantics-aware Dataset Discovery from Data Lakes with Contextualized Column-based Representation Learning. *Proc. VLDB Endow.* 16, 7 (2023), 1726–1739. https://doi.org/10.14778/3587136.3587146

[16] Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, and Michael Stonebraker. 2018. Aurum: A Data Discovery System. In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*. IEEE Computer Society, 1001–1012. https://doi.org/10.1109/ICDE.2018.00094

[17] Yufang Hou, Charles Jochim, Martin Gleize, Francesca Bonin, and Debasis Ganguly. 2019. Identification of tasks, datasets, evaluation metrics, and numeric scores for scientific leaderboards construction. In *Proceedings of the 57th annual meeting of the association for computational linguistics*. 5203–5213.

[18] Xuming Hu, Shen Wang, Xiao Qin, Chuan Lei, Zhengyuan Shen, Christos Faloutsos, Asterios Katsifodimos, George Karypis, Lijie Wen, and Philip S. Yu. 2023. Automatic Table Union Search with Tabular Representation Learning. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*. Association for Computational Linguistics, 3786–3800. https://aclanthology.org/2023.findings-acl.233

[19] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.

[20] Salomon Kabongo, Jennifer D'Souza, and Sören Auer. 2024. ORKG-Leaderboards: a systematic workflow for mining leaderboards as a knowledge graph. *International Journal on Digital Libraries* 25, 1 (2024), 41–54.

[21] Marcin Kardas, Piotr Czapla, Pontus Stenetorp, Sebastian Ruder, Sebastian Riedel, Ross Taylor, and Robert Stojnic. 2020. Axcell: Automatic extraction of results from machine learning papers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 8580–8594.

[22] Aamod Khatiwada, Roee Shraga, Wolfgang Gatterbauer, and Renée J Miller. 2022. Integrating data lake tables. *Proceedings of the VLDB Endowment* 16, 4 (2022).

[23] Aamod Khatiwada, Roee Shraga, and Renée J. Miller. 2023. DIALITE: Discover, Align and Integrate Open Data Tables. In *Companion of the 2023 International Conference on Management of Data, SIGMOD/PODS 2023, Seattle, WA, USA, June 18-23, 2023*, Sudipto Das, Ippokratis Pandis, K. Selçuk Candan, and Sihem Amer-Yahia (Eds.). ACM, 187–190. https://doi.org/10.1145/3555041.3589732

[24] Aamod Khatiwada, Roee Shraga, and Renée J. Miller. 2026. Fuzzy Integration of Data Lake Tables. In *Proceedings 29th International Conference on Extending Database Technology, EDBT 2026, Tampere, Finland, March 24-27, 2026*, Wolfgang Lehner, Vanessa Braganholo, Kostas Stefanidis, Zheying Zhang, Alexander Krause, and João Felipe Nicolaci Pimentel (Eds.). OpenProceedings.org, 96–102. https://doi.org/10.48786/EDBT.2026.08

[25] Seongchan Kim, Keejun Han, Soon Young Kim, and Ying Liu. 2012. Scientific table type classification in digital library. In *Proceedings of the 2012 ACM symposium on Document engineering*. 133–136.

[26] Keti Korini, Ralph Peeters, and Christian Bizer. 2022. SOTAB: The WDC Schema.org table annotation benchmark. In *CEUR Workshop Proceedings*, Vol. 3320. RWTH Aachen, 14–19.

[27] Aristotelis Leventidis, Martin Pekár Christensen, Matteo Lissandrini, Laura Di Rocco, Katja Hose, and Renée J. Miller. 2024. A Large Scale Test Corpus for Semantic Table Search. In *ACM SIGIR*, Grace Hui Yang, Hongning Wang, Sam Han, Claudia Hauff, Guido Zuccon, and Yi Zhang (Eds.). ACM, 1142–1151. https://doi.org/10.1145/3626772.3657877

[28] Aristotelis Leventidis, Laura Di Rocco, Wolfgang Gatterbauer, Renée J. Miller, and Mirek Riedewald. 2023. DomainNet: Homograph Detection and Understanding in Data Lake Disambiguation. *ACM Trans. Database Syst.* 48, 3 (2023), 9:1–9:40. https://doi.org/10.1145/3612919

[29] Ziyu Li, Henk Kant, Rihan Hai, Asterios Katsifodimos, Marco Brambilla, and Alessandro Bozzon. 2023. Metadata representations for queryable repositories of machine learning models. *IEEE Access* 11 (2023), 125616–125630.

[30] Ziyu Li, Hilco Van Der Wilk, Danning Zhan, Megha Khosla, Alessandro Bozzon, and Rihan Hai. 2024. Model Selection with Model Zoo via Graph Learning. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 1296–1309.

[31] Weixin Liang, Nazneen Rajani, Xinyu Yang, Ezinwanne Ozoani, Eric Wu, Yiqun Chen, Daniel Scott Smith, and James Zou. 2024. Systematic analysis of 32,111 AI model cards characterizes documentation practice in AI. *Nature Machine Intelligence* 6, 7 (2024), 744–753.

[32] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python Toolkit for Reproducible Information Retrieval Research with Sparse and Dense Representations. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*. 2356–2362.

[33] Jimmy Lin and Pengyi Zhang. 2007. Deconstructing nuggets: the stability and reliability of complex question answering evaluation. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. 327–334.

[34] Jiarui Liu, Wenkai Li, Zhijing Jin, and Mona T. Diab. 2024. Automatic Generation of Model and Data Cards: A Step Towards Responsible AI. In *ACL*, Kevin Duh, Helena Gómez-Adorno, and Steven Bethard (Eds.). Association for Computational Linguistics, 1975–1997. https://doi.org/10.18653/V1/2024.NAACL-LONG.110

[35] Daohan Lu, Sheng-Yu Wang, Nupur Kumari, Rohan Agarwal, Mia Tang, David Bau, and Jun-Yan Zhu. 2023. Content-based search for deep generative models. In *SIGGRAPH Asia 2023 Conference Papers*. 1–12.

[36] Shaoyin Ma, Chenggong Hu, Huiqiong Wang, Li Sun, Mingli Song, and Jie Song. 2025. HuggingR^4: A Progressive Reasoning Framework for Discovering Optimal Model Companions. *arXiv preprint arXiv:2511.18715* (2025).

[37] Songzhu Mei, Cong Liu, Qinglin Wang, and Huayou Su. 2022. Model Provenance Management in MLOps Pipeline. In *Proceedings of the 2022 8th International Conference on Computing and Data Engineering*. Association for Computing Machinery, New York, NY, USA, 45–50. https://doi.org/10.1145/3512850.3512861

[38] Renée J. Miller. 1998. Using Schematically Heterogeneous Structures. In *SIGMOD*. ACM Press, 189–200. https://doi.org/10.1145/276304.276322

[39] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. Model Cards for Model Reporting. In *Proceedings of the Conference on Fairness,*

Accountability, and Transparency. Association for Computing Machinery, New York, NY, USA, 220–229. https://doi.org/10.1145/3287560.3287596

[40] Alistair Moffat and Justin Zobel. 2008. Rank-biased precision for measurement of retrieval effectiveness. *ACM Transactions on Information Systems (TOIS)* 27, 1 (2008), 1–27.

[41] Xin Mu, Yu Wang, Yehong Zhang, Jiaqi Zhang, Hui Wang, Yang Xiang, and Yue Yu. 2023. Model Provenance via Model DNA. arXiv:2308.02121 [cs.LG]

[42] Fatemeh Nargesian, Erkang Zhu, Renée J. Miller, Ken Q. Pu, and Patricia C. Arocena. 2019. Data Lake Management: Challenges and Opportunities. *Proc. VLDB Endow.* 12, 12 (2019), 1986–1989. https://doi.org/10.14778/3352063.3352116

[43] Fatemeh Nargesian, Erkang Zhu, Ken Q. Pu, and Renée J. Miller. 2018. Table Union Search on Open Data. *Proc. VLDB Endow.* 11, 7 (2018), 813–825. https://doi.org/10.14778/3192965.3192973

[44] Ani Nenkova and Rebecca J Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *Proceedings of the human language technology conference of the north american chapter of the association for computational linguistics: Hlt-naacl 2004*. 145–152.

[45] Koyena Pal, David Bau, and Renée J. Miller. 2025. Model Lakes. In *EDBT*. OpenProceedings.org, 985–995.

[46] Wei Pang, Xiangru Jian, Hehan Li, Zhixuan Yu, Alex Xue, Jinyang Li, Zhengyuan Dong, Xinjian Zhao, Hao Xu, Chao Zhang, et al. 2026. TRL-Bench: Standardizing Cross-Paradigm Representation-Level Evaluation of Tabular Encoders. *arXiv preprint arXiv:2606.09323* (2026).

[47] Ronak Pradeep, Nandan Thakur, Shivani Upadhyay, Daniel Campos, Nick Craswell, and Jimmy Lin. 2024. Initial nugget evaluation results for the trec 2024 rag track with the autonuggetizer framework. *arXiv preprint arXiv:2411.09607* (2024).

[48] Ronak Pradeep, Nandan Thakur, Shivani Upadhyay, Daniel Campos, Nick Craswell, Ian Soboroff, Hoa Trang Dang, and Jimmy Lin. 2025. The great nugget recall: Automating fact extraction and rag evaluation with large language models. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 180–190.

[49] 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)*. 3982–3992.

[50] Furkan Şahinuç, Thy Thy Tran, Yulia Grishina, Yufang Hou, Bei Chen, and Iryna Gurevych. 2024. Efficient performance tracking: Leveraging large language models for automated construction of scientific leaderboards. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. 7963–7977.

[51] Saron Samuel, Andrew Yates, Dawn Lawrie, Ian Soboroff, Trevor Adriaanse, Benjamin Van Durme, and Eugene Yang. 2026. CoverageBench: Evaluating Information Coverage across Tasks and Domains. *arXiv preprint arXiv:2603.20034* (2026).

[52] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*. Vol. 39. Cambridge University Press Cambridge.

[53] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems* 36 (2023), 38154–38180.

[54] Roee Shraga and Renée J. Miller. 2023. Explaining Dataset Changes for Semantic Data Versioning with Explain-Da-V. *Proc. VLDB Endow.* 16, 6 (2023), 1587–1600. https://doi.org/10.14778/3583140.3583169

[55] Shruti Singh, Shoaib Alam, Husain Malwat, and Mayank Singh. 2024. Legobench: Scientific leaderboard generation benchmark. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. 14598–14613.

[56] The ModelScope Team. 2023. ModelScope: bring the notion of Model-as-a-Service to life. https://github.com/modelscope/modelscope.

[57] Ellen M Voorhees et al. 1999. The TREC-8 question answering track report. In *Trec*, Vol. 99. 77–82.

[58] Keyu Wang, Abdullah Norozi Iranzad, Scott Schaffter, Doina Precup, and Jonathan Lebensold. 2024. Mitigating Downstream Model Risks via Model Provenance. arXiv:2410.02230 [cs.LG] https://arxiv.org/abs/2410.02230

[59] Jian Wu, Jiayu Zhang, Dongyuan Li, Linyi Yang, Aoxiao Zhong, Renhe Jiang, Qingsong Wen, and Yue Zhang. 2025. League: Leaderboard Generation on Demand. *arXiv preprint arXiv:2502.18209* (2025).

[60] Sean Yang, Chris Tensmeyer, and Curtis Wigington. 2022. Telin: Table entity linker for extracting leaderboards from machine learning publications. In *Proceedings of the first Workshop on Information Extraction from Scientific Publications*. 20–25.

[61] ChengXiang Zhai, William W Cohen, and John Lafferty. 2015. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *Acm sigir forum*, Vol. 49. ACM New York, NY, USA, 2–9.

[62] Erkang Zhu, Fatemeh Nargesian, Ken Q. Pu, and Renée J. Miller. 2016. LSH Ensemble: Internet-Scale Domain Search. *Proc. VLDB Endow.* 9, 12 (2016), 1185–1196. https://doi.org/10.14778/2994509.2994534

[63] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*. 22–32.