Renaming and Retrieval: Semantic Sensitivity in Join Discovery on a YADL-Augmented Data Lake

Rosina Kharal
David R. Cheriton School of Computer Science, University of Waterloo
rkharal@uwaterloo.ca

Abstract

Table retrieval in data lakes often assumes that table and column names are stable and useful. This paper tests that assumption using two semantic variants of the YADL binary data lake [1]. The first variant, *Join-Preserving*, renames predicates but keeps all rows unchanged. The second variant, *Challenge*, renames predicates and reduces row overlap to about 22.5% of the original table.

Across five retrieval methods, value-overlap methods are robust to renaming and only mildly affected by weaker overlap. In contrast, *starmie-schema* is sensitive to semantic renaming, but it does not reliably identify tables that are actually joinable. We therefore test *weighted starmie*, a hybrid method that combines schema similarity with a small sample of 32 column values. This method improves real top-1 containment by about 230× over schema-only retrieval while keeping strong ranking confidence. These results suggest that semantic names are useful, but they should be combined with value evidence when the goal is join discovery.

CCS Concepts

• **Information systems** → **Data mining**; *retrieval methods, join candidates.*

Keywords

data lakes, table discovery, join candidate retrieval, semantic table matching, column embeddings

Artifact: https://github.com/rkharal/YADL-Semantic-Extensions

1 Introduction

Join retrieval is an emerging area of study, driven by the massive amounts of data now stored in data warehouses called data lakes. The format of this data is inconsistent at best with all of many of the following characteristics: (i) mixed file formats, (ii) various types of metadata, (iii) missing column names, and (iv) with or without null-value cleaning. Data is ingested into these massive storage systems in high volumes and at high speed, and it becomes the task of the data scientist to understand the data and find information useful for training a model. Our problem setting begins with a query, in the form of a query table. A search is conducted through the data lake to find good candidate tables for join operations; this will augment the query table and provide a greater number of features for the training data. The goal is more reliable and helpful features for better long-term predictions in the machine learning task.

The study by Cappuzzo et al., which examines the retrieve-merge-predict pipeline [1], created a semi-synthetic benchmark data lake, YADL, for exactly this purpose. A need existed for a data lake that is deterministic, where join relationships and their outcomes are predictable, such that relevant join candidates could be better distinguished from unrelated, poor ones.

The *Retrieve-Merge-Predict* study evaluates exact matching for the selection of join candidates, where selection depends on a high degree of value overlap between the query column and the candidate column. The table or column name itself, as an attribute or as metadata in the column header, is not examined by the retrieval approaches used in that study. Further study and design of YADL was needed to evaluate semantic relationships between a query column and a join candidate column.

In the study of data lakes and reliable join candidates, the promise of semantic retrieval and more naming-aware approaches is often discussed, but not well studied. Its impact, positive or negative, is not well quantified in this area [3–5]. Therefore, this study hopes to add clarity and bridge some of the gaps in this area.

This study is a step toward better understanding the semantic relationships that exist between a query column and a join candidate column. We hope to determine whether semantic retrieval is a viable and reasonable path forward in the process of table joins for feature engineering from data lakes.

The Retrieve-Merge-Predict paper introduces YADL and evaluates retrieval, merge, selection aggregation, and prediction strategies for query table augmentation. It emphasizes that the *quality* of retrieved candidate tables affects the full pipeline, and that resource cost must be weighed alongside prediction performance. The retrieval methods evaluated in that paper primarily rely on exact-match column overlap; Starmie is the one exception. Starmie is a semantic dataset-discovery method that learns contextualized column embeddings through contrastive learning. It represents columns as vectors and compares them using embedding similarity, which makes it relevant in cases where columns are semantically related but not exactly identical. In the context of the Retrieve-Merge-Predict paper, Starmie showed no benefit over exact matching. Starmie indicated no measurable loss, but added additional time required to build embedding vectors and run the contrastive-learning step.

Joining a base query table against a large, schema-less data lake requires first finding which tables in the lake are worth joining. This is the retrieval stage of the pipeline. The findings by Cappuzzo et al. [1] illustrate that simple value-overlap retrieval, measured by Jaccard containment, performs close to optimally against a costlier semantic embedding-based alternative (Starmie [8]). That comparison, however, is run entirely on a lake where column and file names come directly from a single, consistent knowledge base (YAGO). Real data lakes are rarely this well constructed: the same column quantities are often exported, copied, or renamed under a different label by a different team. Our study asks *what happens to each*

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

retrieval class when naming is deliberately disturbed, does the underlying data change along with it. We experiment with a low-cost hybrid approach, in the hope of bridging the gap between exact-matching and semantic retrieval methods.

2 Contributions

In this work, we construct two controlled semantic variants of YADL Binary through column and table renaming:

- *YADL_Binary_Semantic_Join-Preserving*: construct a variant that renames each predicate column with semantic alternatives while preserving row-level overlap exactly. This isolates the effect of naming alone.
- *YADL_Binary_Semantic_Challenge*: construct a harder variant that renames predicate columns and additionally weakens row-level overlap to roughly 22.5% of the source table. This allows studying naming and overlap loss together, rather than in isolation.
- Evaluate five retrieval methods: `Exact Matching`, `MinHash`, `Hybrid MinHash`, `Starmie-values`, and a schema only variant we call `Starmie-schema`. For both variants, we use top-1 score, score decay, semantic copy share, and canonical overlap as complementary metrics.
- Show that value-overlap methods are robust to renaming, while `Starmie-schema` reliably detects renamed tables but is a poor predictor of whether a retrieved table is actually a good join candidate.
- Introduce `Weighted Starmie`, a lightweight hybrid approach that combines a trimmed value-embedding signal with schema similarity, and show that it substantially improves real join candidate quality over schema-only retrieval with much less cost of full-column embeddings.
- Summarize these results into concrete, practical recommendations for retrieval design in data lakes where naming cannot be assumed to be stable or informative.

3 Background

3.1 YADL Binary and the Retrieve-Merge-Predict Pipeline

In their work, Cappuzzo et al. designed YADL, a semi-synthetic data lake built from the YAGO knowledge base [10, 11]. Its simplest variant, *YADL Binary*, simply reshapes every YAGO predicate into a two-column table with columns `subject` and the predicate name. For example, the triplet `(France, hasCapital, Paris)` becomes a row of the `binary-hasCapital.parquet` file. The parquet file is considered to be the name of the table. YADL Binary contains 70 such tables, 20099403 rows, and 140 columns in total, and is the base dataset we build on throughout this paper. Cappuzzo et al. evaluate retrieval as the first of three pipeline stages (retrieve, merge, predict) and show that retrieval quality is strongly correlated with high Jaccard containment (JC), described with an example below. High JC has a positive impact on the downstream prediction task because it minimizes the missing values introduced by imperfect joins. Our experiments cover the retrieval stage, and specifically the impact of semantic renaming on retrieval methods that (a) use exact matching only, and (b) apply some degree of semantic awareness in the retrieval task.

3.2 Jaccard Containment

Jaccard containment measures how much of a query column's content is recoverable from a candidate column, and it is the similarity measure that `Exact Matching`, `MinHash`, and `Hybrid MinHash` all rely on. Given a query column Q and a candidate column C, both treated as sets of values, the Jaccard containment of C within Q is

$$\mathrm{JC}(Q, C) = \frac{|Q \cap C|}{|Q|}. \tag{1}$$

The numerator is the number of values shared between the two columns, and the denominator is the size of the query column alone. Note, we are not taking the union of both columns, which is what distinguishes *containment* from the more familiar Jaccard *similarity*. This distinction matters in practice: a query column can be fully contained within a much larger candidate column and still receive a containment score of 1.0, even though the candidate column also holds many values the query never asked about. Containment is also asymmetric: $\mathrm{JC}(Q, C)$ is generally *not equal* to $\mathrm{JC}(C, Q)$, since swapping which column plays the role of Q changes the denominator.

As a simple example, suppose a query column Q contains the three countries {`France`, `Italy`, `Japan`}, and a candidate column C contains the four countries {`France`, `Italy`, `Germany`, `Spain`}. The intersection $Q \cap C$ is {France, Italy}, so $|Q \cap C| = 2$ and $|Q| = 3$, giving

$$\mathrm{JC}(Q, C) = \frac{2}{3} \approx 0.667. \tag{2}$$

In other words, about 66.7% of the query column's values were found in the candidate column, even though the candidate column contains two additional countries that have nothing to do with the query. A retrieval method built on Jaccard containment would rank this candidate favorably on the strength of that 0.667 score, regardless of what `Germany` and `Spain` happen to be doing in the same column.

4 Related Work

A separate line of work, mostly outside the data-lake literature, has studied what happens when a table or database schema is deliberately renamed with synonyms. Gan et al. [9] introduced Spider-Syn, a human-curated variant of the Spider text-to-SQL benchmark in which schema-related words in natural language questions are replaced with manual synonyms that mimic how real users may paraphrase queries (for example, `country` becomes `nation`). They found that this single substitution, without any adversarial intent, caused a dramatic drop in accuracy for state-of-the-art text-to-SQL models, because those models rely heavily on lexical matching between question words and schema tokens. Chang et al. [2] later generalized this idea in Dr.Spider, a diagnostic benchmark that applies seventeen different varying perturbations across databases, questions, and queries. One of the seventeen specifically renamed database columns with synonyms or abbreviations. They report that even the most robust model they tested suffered a substantial

overall performance drop, reinforcing that synonym renaming is not a minor edge case but a genuine failure mode.

Within the data-lake literature specifically, Dong et al. [6] proposed PEXESO, a framework for discovering joinable tables in data lakes that explicitly incorporates synonym and abbreviation matching, rather than relying on literal value overlap alone, such that a column named `nation` could still be matched against a column named `country`. Their follow-up work, DeepJoin [7], improved on PEXESO by further building on a pretrained language model specifically for the joinability task. This involved embedding each column as text and ranking candidates by embedding similarity, and showed that this learned approach found substantially more true joins than value-overlap methods on columns where names and values had both been altered. Both papers demonstrate, from within the data-lake retrieval literature itself, that name-aware matching can recover joinable columns that exact or approximate value-overlap methods miss entirely.

However, none of them isolate naming from row-level value overlap as two independently controllable variables as we test in this work.

5 Methods

We compare four methods evaluated in the original YADL study, plus two Starmie variants we construct for this study.

5.1 Value-Overlap Retrieval Methods

Exact Matching computes Jaccard containment: $\mathrm{JC}(Q, C) = |Q \cap C|/|Q|$ between a query column Q and every candidate column C in the lake, ranking by this score directly. It needs no offline preparation beyond loading the full data lake, but its query cost scales with the number of columns scanned.

MinHash builds an *offline locality-sensitive-hashing* index [12] over the data lake and, at query time, returns any candidate whose estimated containment exceeds a fixed threshold. Results are not ranked. It is fast to query but sacrifices both ranking quality and top-k rankings.

Hybrid MinHash shortlists candidates with the MinHash index and then re-ranks that shortlist using exact containment, recovering most of `Exact Matching`'s ranking quality while scanning far fewer columns than a full pass over the lake as in `Exact Matching`.

5.2 Starmie-Based Semantic Retrieval Methods:

Before describing our specific configurations, it is worth briefly stating what an embedding is, since three of our five retrieval methods depend on one. An *embedding* is a fixed-length vector produced by a neural network, such that inputs judged similar in meaning land close together in the vector space. A column from the data lake is taken and its text is serialized. This is irrespective of whether that text comes from values, from names, or from both. The serialization is then tokenized, passed through the encoder, and pooled into a single vector. Next, two columns are compared by the cosine of the angle between their vectors. A cosine similarity near 1 indicates the two texts are close together in the model's learned space; it is not a percentage or an accuracy figure, and it is only meaningful and comparable within a single method.

Starmie-based semantic retrieval. We use a lightweight Starmie type retriever rather than the full contrastive trained Starmie pipeline. In our implementation, each candidate column is serialized as text, embedded with a pretrained transformer encoder, and ranked by cosine similarity to the serialized query-column embedding. This simplified setup lets us compare value-based and schema-name-based semantic signals without introducing an additional contrastive training stage.

Starmie-values. In the value-based configuration, the serialized text consists of a sample of cell values from the candidate column. For each candidate column, we use up to 128 distinct non-null values, and we serialize the query column in the same way. This configuration measures whether semantic embeddings over column contents can retrieve useful join candidates.

Starmie-schema. We also evaluate a schema-only configuration, called *Starmie-schema*, in which the serialized text contains only the candidate table filename and the candidate column name. No cell values are included. For example, a candidate may be serialized as `Table: gross_income, columns subject: Gross Income`. Thus, *Starmie-schema* uses zero rows from the candidate table. This lightweight configuration isolates naming sensitivity: it tests whether the embedding model notices semantic predicate renamings independently of any signal from overlapping cell values.

Weighted Starmie. Toward the end of our study, we also evaluate a hybrid variant, *Weighted Starmie*, designed to combine the strongest signals from *Starmie-values* and *Starmie-schema*. The motivation is that value embeddings capture row-content similarity, while schema embeddings capture semantic information from table and column names. The weighted variant uses both signals instead of treating them separately. In the lightweight value embeddings, we use only 32 value.

For each candidate column, we construct two serialized representations. The first is a value representation, built from a sample of the column's cell values. The second is a schema representation, built from the candidate table filename and candidate column name. We embed both representations with the same pretrained transformer encoder and compute cosine similarity against the corresponding query-column embeddings. The final retrieval score is a weighted combination:

$$s(q, c) = 0.6 \cdot s_{value}(q, c) + 0.4 \cdot s_{schema}(q, c), \tag{3}$$

where s_{value} is the value-based cosine similarity and s_{schema} is the schema-name-based cosine similarity. In this configuration, we use up to 32 distinct non-null cell values per column for the value representation. The schema representation uses no cell values; it contains only table and column names. Thus, `Weighted Starmie` remains lighter than the full value-only setting while still incorporating row-content evidence alongside naming evidence.

6 Constructing Semantic Variants

Both variants keep all 70 original tables unmodified and add two synthetic *semantic tables* per original (140 new tables, 210 total), with every synthetic table traceable back to its source predicate through an audit file.

Join-Preserving. Each semantic copy is byte-for-byte identical to its source in every row; only the file name and the predicate column header are rewritten (for example, `binary-hasCapital.parquet` becomes `seat_of_government.parquet`, with `hasCapital` renamed to `Seat of Government`). The resulting lake has 210 tables, 40198806 semantic rows, and occupies 804MB, with 100% overlap between every semantic table and its source, by construction.

Challenge. Each semantic table is built to roughly 0.75× its source's row count, with about 30% of its own rows sampled from the true source predicate and the remaining 70% donated from a related predicate group (a Challenge copy of `hasCapital` draws from `isLocatedIn`, `hasNeighbor`, `hasArea`, and similar). Composing these ratios, $0.75 \times 0.30 = 0.225$, means only about 22.5% of a source table's rows survive in each Challenge copy. The resulting lake has the same 210 tables, 30149096 semantic rows, and occupies 702MB.

We evaluate all methods against the same seven query cases in the Cappuzzo study [1], used throughout the Join-Preserving and Challenge experiments (spanning elections, housing, employment, movies, accidents, and population data), reporting top-1 score, semantic copy share (the fraction of a top-k list that is a newly added table), and canonical overlap (semantic copies mapped back to their source before scoring, the cleanest measure of whether a method still finds the correct predicate).

7 Experiments & Results

7.1 Join-Preserving: A Sanity Check

We first evaluate retrieval on the Join-Preserving semantic lake. This setting is a form of a sanity check: every original YADL Binary table is kept unchanged, and two renamed semantic copies are added for each table. The rows in the semantic copies are identical to the original rows, so value-based retrieval methods should remain stable, and schema-based methods may change because the table and column names have changed.

Table 1 reports mean top-1 retrieval scores on the original YADL Binary lake and the Join-Preserving lake. Exact Matching and Hybrid MinHash have identical mean top-1 scores in both settings, confirming that renaming tables and columns does not affect methods based on cell-value overlap. `Starmie-values` changes only slightly (+0.0029), which is consistent with small variation from embedding sampled values rather than a substantive retrieval change. `Starmie-schema` shows the largest shift (+0.0235), as expected, because it uses table and column names as its retrieval signal. Figure 1 shows that this pattern is consistent across query cases.

Canonical overlap provides a complementary view of the retrieval behaviour. For each method, we map every renamed semantic table back to its original YADL binary predicate and compare the augmented-lake top-k list with the original-lake top-k list. Thus, canonical overlap measures whether a method retrieves the same underlying predicate families after semantic variants are added.

In Join-Preserving, the semantic copies preserve the same rows as their source tables. This creates many tied or near-tied candidates for value-based methods. For example, an original predicate and its two semantic copies can all receive the same containment score, causing multiple versions of the same predicate family to occupy the top-k list. `Hybrid MinHash` and `MinHash` recover the same

Table 1: Join-Preserving: mean top-1 score, original lake vs. Join-Preserving. MinHash reports the LSH threshold label used for retrieval, not a calibrated similarity score, and should not be compared directly with the other score values.

Method	Original	JoinPreserving	Delta
Exact Matching	0.9569	0.9569	0.0000
MinHash	20.0000	20.0000	0.0000
Hybrid MinHash	0.8568	0.8568	0.0000
Starmie-values	0.9754	0.9783	0.0029
Starmie-schema	0.8533	0.8768	0.0235

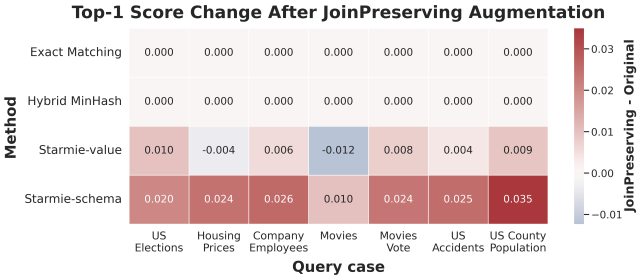

Figure 1: Per-query-case top-1 delta under Join-Preserving. Value-based methods have zero or near-zero change; `Starmie-schema` changes consistently because it is sensitive to the renamed semantic table and column names.

Table 2: Canonical overlap at $k = 30$ and semantic copy share at $k = 10$ for both semantic variants. Canonical overlap measures how much the augmented-lake top-30 retrieval list agrees with the original-lake top-30 list after renamed semantic copies are mapped back to their original YADL binary predicates. Copy share is the fraction of each method's top-10 list made up of newly added semantic tables.

	Canonical @30		Copy Share @10	
Method	JP	Ch.	JP	Ch.
Exact Matching	0.3965	0.3900	64.3%	52.9%
Hybrid MinHash	1.0000	0.9464	70.0%	65.7%
MinHash	1.0000	0.9464	68.6%	72.9%
Starmie-values	0.4619	0.4985	62.9%	64.3%
Starmie-schema	0.3095	0.3179	94.3%	97.1%

underlying predicate families at rank 30 (1.0000 canonical overlap), whereas `Starmie-schema` achieves only 0.3095. This indicates that `Starmie-schema` reacts strongly to the new semantic names, but its top-30 ranking is much less aligned with the original-lake ranking.

Table 2 reports canonical overlap at $k = 30$ together with semantic copy share at $k = 10$. Copy share measures the fraction of each method's top-10 list made up of newly added semantic tables rather than original `binary` tables. `Starmie-schema` has the highest copy share in both variants (94.3% in Join-Preserving and 97.1% in Challenge), showing that renamed semantic tables dominate its highest-ranked recommendations more than they do for value-based methods.

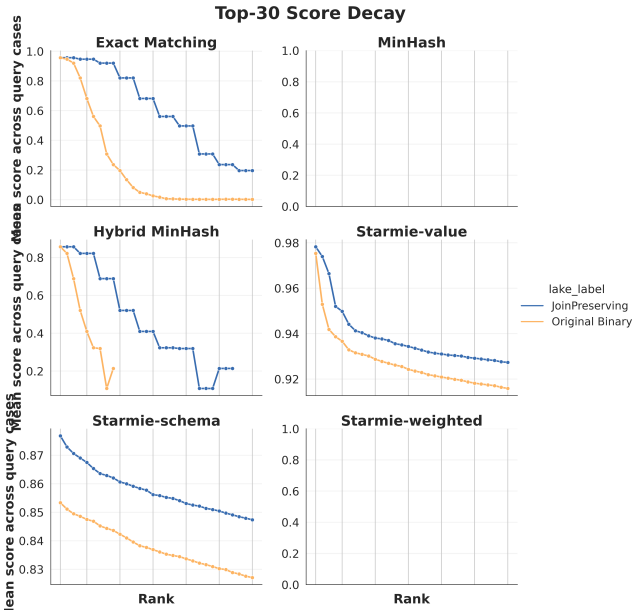

Figure 2: Top-30 score decay under Join-Preserving. Because semantic copies preserve the original rows, value-based methods produce tied or near-tied high-scoring candidates, causing their augmented-lake curves to decay more slowly than the original-lake curves. `Starmie-schema` also scores the augmented lake higher across ranks, reflecting its sensitivity to semantic table and column names.

Figure 2 shows how retrieval scores decay across the top 30 ranks under Join-Preserving. For `Exact Matching` and `Hybrid MinHash`, the Join-Preserving curves decay more slowly than the original-lake curves because the added semantic copies preserve the same rows and therefore cluster at similar containment levels. `Starmie-schema` also assigns consistently higher scores in the augmented lake, reflecting its sensitivity to the added semantic table and column names.

7.2 Challenge: A Harder Test

Next, we evaluate retrieval on the Challenge semantic lake. Unlike Join-Preserving, the Challenge variant does not create exact row copies. Each original YADL Binary table is still kept unchanged, but the two added semantic variants are blended tables where each variant is about 0.75 times the size of the original table, and about 22.5% of the variant rows come from the original predicate, and the remaining rows come from related donor predicates. Thus, Challenge tests whether retrieval behaviour changes when semantic names are present but exact value overlap is weaker.

Table 3 compares mean top-1 retrieval scores on the original lake and the Challenge lake. `Exact Matching` has a small positive delta (+0.0066), indicating that donor rows can sometimes introduce additional incidental value matches. `Hybrid MinHash` has a small negative mean delta (−0.0100), but Figure 3 shows that this is not a uniform degradation across all queries; it is driven primarily by the `housing_prices` query case. `Starmie-value` changes only slightly

Table 3: Challenge: mean top-1 score, original lake vs. Challenge. MinHash reports the LSH threshold label used for retrieval, not a calibrated similarity score, and should not be compared directly with the other score values.

Method	Original	Challenge	Delta
Exact Matching	0.9569	0.9635	+0.0066
MinHash	20.0000	20.0000	0.0000
Hybrid MinHash	0.8568	0.8468	−0.0100
Starmie-value	0.9535	0.9559	+0.0024
Starmie-schema	0.8520	0.8765	+0.0245

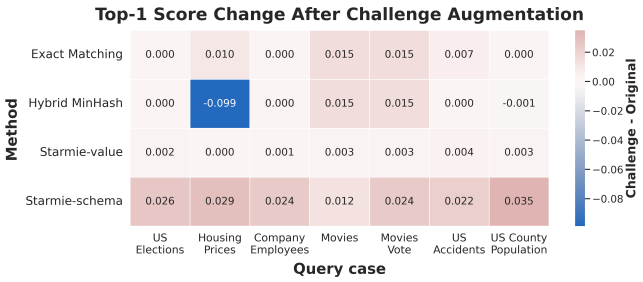

Figure 3: For each query-case top-1 delta under Challenge. The heatmap shows whether the mean changes in Table 3 are systematic or query-specific. `Hybrid MinHash`'s negative mean is driven mainly by the *housing_prices* case, while `starmie-schema` shows consistently positive deltas because it is sensitive to the added semantic table and column names.

(+0.0024), suggesting that its value-embedding signal is relatively stable under the Challenge augmentation. `Starmie-schema` again shows the largest positive shift (+0.0245), which is expected because it uses only table and column names and is therefore sensitive to the newly added semantic labels rather than to the amount of row overlap.

Figure 3 breaks the mean deltas down by query case. This figure is important because the averages in Table 3 can hide whether a method changes consistently or whether the mean is driven by a small number of cases. `Exact Matching` is mostly stable, with small positive changes where donor rows create additional overlap. `Hybrid MinHash` is also mostly stable, except for the negative *housing_prices* cell, which accounts for its negative mean. `Starmie-schema` is positive across the query cases because the Challenge lake contains many new semantically named tables, and this method ranks candidates from schema text rather than from only row overlap. These results confirm expectations.

7.3 Weighted Starmie: Combining Names and Values

The previous experiments show that `starmie-schema` is highly sensitive to renamed semantic tables, but schema similarity alone is not enough to identify useful joins. A table can have a semantically relevant name while still having little or no overlap with the query column. To test whether semantic retrieval can be made more useful

Table 4: `Weighted Starmie` vs. `starmie-schema` on the Challenge lake. Actual containment measures literal value overlap with the query column and is independent of the method's embedding scores.

Metric	Starmie-schema	Starmie-weighted
Mean top-1 score	0.8765	0.9151
Mean top-1 actual containment	0.0006	0.1400
Copy share @10	97.14%	68.57%

for downstream joining, we evaluate a weighted starmie score on the harder Challenge lake.

The weighted score combines value similarity and schema similarity:

$$\text{score} = 0.6 \times \text{sim}_{\text{value}} + 0.4 \times \text{sim}_{\text{schema}}. \quad (4)$$

Here, $\text{sim}_{\text{value}}$ is computed from at most 32 sampled values per column, making the method much lighter than embedding full columns, while $\text{sim}_{\text{schema}}$ is computed from table and column names. The purpose of this experiment is to test whether adding a value-based signal helps distinguish candidates that are merely semantically named from candidates that are actually useful for joins.

We compare `starmie-weighted` against `starmie-schema` on the Challenge lake and report *actual containment*: the literal overlap between the query column and the retrieved candidate column. This metric is independent of either method's internal embedding score and directly measures whether a retrieved candidate is likely to support a useful join.

Table 4 summarizes the result. Starmie-weighted substantially improves join usefulness: its mean top-1 actual containment is 0.1400, compared with only 0.0006 for `starmie-schema`. Its mean top-1 retrieval score is also higher (0.9151 vs. 0.8765), indicating that the improvement in containment does not come from lower-confidence rankings. The trade-off is that `Starmie-weighted` retrieves fewer semantic copies at $k = 10$ (68.57% vs. 97.14%). This is desirable for *retrieve-merge-predict*: the goal is not only to find renamed semantic tables, but to find candidates that are also likely to join.

Figure 4 shows the containment improvement by query case. `Starmie-schema` often retrieves renamed tables with almost no actual overlap, while `starmie-weighted` retrieves candidates with much stronger containment in several cases. For example, for the `movies` query, `starmie-weighted` retrieves `helmed_by.parquet` (`Helmed By`) as its top-1 candidate with actual containment 0.6226, while `starmie-schema`'s top-1 candidate for the same query has containment 0.0000. This illustrates the main advantage of the weighted approach: it preserves semantic sensitivity while filtering candidates through value evidence.

Figure 5 decomposes the weighted score across the top 10 ranks. The value component is generally stronger than the schema component, and the final weighted score follows the value component closely. This confirms that the weighted score is not just reproducing schema-only behaviour; it is using value similarity to improve the quality of retrieved join candidates.

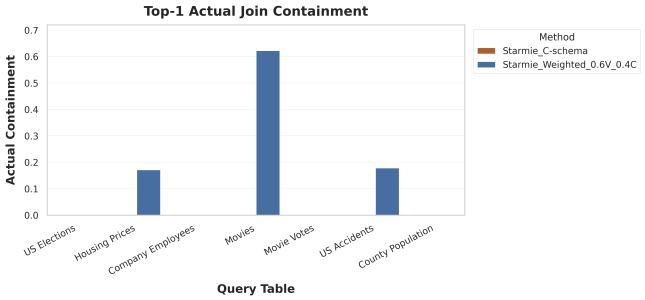

Figure 4: Top-1 actual containment by query case on the Challenge lake. `Starmie-schema` often retrieves semantically named candidates with little or no literal overlap, while `Starmie-weighted` retrieves candidates with substantially stronger join overlap in several query cases.

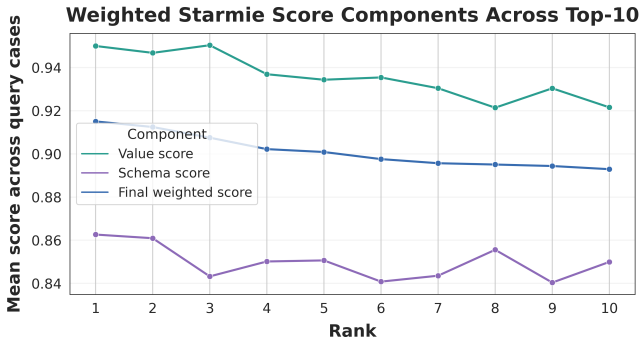

Figure 5: `Weighted Starmie` score decomposition across the top 10 ranks. The final score combines value similarity and schema similarity, but the curve follows the value component closely, showing that value evidence drives much of the improved ranking behaviour.

8 Discussion and Recommendations

This study examined how semantic table and column names affect join-candidate retrieval in YADL Binary. We created two semantic variants of the original binary data lake. In Join-Preserving, each original table was kept unchanged and two renamed semantic copies were added with identical rows. This setting tests whether semantic names change retrieval behaviour when joins remain strong. In Challenge, the original tables were again kept unchanged, but the semantic copies were blended with donor rows from related predicates. This setting tests whether semantic names still help when exact value overlap is weakened, and the added tables become noisier.

The first finding is that value-overlap methods remain strong baselines. `Exact Matching` and `Hybrid MinHash` are largely insensitive to renaming because they compare cell values rather than table or column names. In Join-Preserving, this is the expected behaviour: since the rows are unchanged, value-based scores remain stable. In Challenge, value-based methods are affected by the noisier blended tables, but they still provide an important signal of whether a candidate can actually join with the query table. This

makes value overlap a reliable default when the goal is downstream merge-predict performance.

The second finding is that schema-based retrieval captures a different kind of signal. *Starmie-schema* is sensitive to semantic names and retrieves many of the newly added renamed tables. This confirms that schema text can expose relationships that value-only methods cannot directly see, such as recognizing that `hasCapital` and `Seat of Government` describe related concepts. However, schema sensitivity alone does not guarantee join usefulness. `Starmie-schema` has low canonical overlap and very low actual containment in the weighted-Starmie comparison, showing that a semantically plausible table name can still correspond to a weak join candidate.

The third finding is that combining value and schema signals is more effective than using schema alone. `Starmie-weighted` blends value-embedding similarity with schema-name similarity:

$score = 0.6 \times sim_{value} + 0.4 \times sim_{schema}$

Using only 32 sampled values per column, this method substantially improves actual containment compared with `starmie-schema`. This suggests that schema information is useful, but it should be used together with value evidence. In practice, schema-only retrieval is best understood as a discovery mechanism, while weighted retrieval is better suited for ranking final join candidates.

These results lead to three recommendations. First, do not rely on schema similarity alone when the goal is to find joinable tables. Schema similarity can identify semantically related candidates, but it does not verify that the table has row overlap to support a useful join. Second, use schema-based retrieval as a complementary signal, especially in data lakes where table and column names contain meaningful natural-language descriptions. Third, prefer weighted or hybrid retrieval when table naming is inconsistent or when value overlap may be weak. The weighted approach preserves the semantic benefit of names while using value evidence to improve join reliability.

The study also has limitations. Our experiments evaluate retrieval only, not the full retrieve-merge-predict pipeline. Therefore, we can show that `Starmie-weighted` improves actual containment, but we have not yet shown that this improvement leads to better downstream prediction accuracy. A second limitation is that the query-side tables retain relatively generic column names, while most semantic renaming is applied to the candidate side. This makes the test conservative for schema-based methods. A third limitation is that all experiments use YADL Binary, where every table has only two columns. Wider YADL variants may provide a more indepth schema context and may change the relative benefit of schema-based retrieval.

The next step is to run the full retrieve-merge-predict pipeline. This would test whether the retrieval improvements observed here translate into improved predictive performance after joining candidate tables to the source table. In particular, the key question is whether `starmie-weighted` higher actual containment produces better downstream model accuracy than schema-only retrieval or value-overlap retrieval alone. Future work should also repeat the study on richer YADL variants such as YADL Base, YADL 10k, and YADL 50k, where tables may be wider, noisier, and more realistic. Finally, future experiments should consider symmetric semantic renaming on both the query and candidate sides to give schema-based methods a fairer opportunity to compare meaningful names on both ends of the retrieval task.

9 Conclusion

This study shows that semantic names change join-candidate retrieval behaviour, but that semantic naming alone is not enough to identify useful joins. In Join-Preserving, value-based methods remain stable because the rows are unchanged, while Starmie-schema reacts to the renamed semantic tables. In Challenge, where semantic copies are blended with donor rows and value overlap is weak, schema-based retrieval continues to find renamed tables, but many of these candidates have weak actual join overlap.

The strongest result is that a lightweight weighted approach improves the balance between semantic discovery and join usefulness. `Starmie-weighted` combines schema-name similarity with value-embedding similarity from only 32 sampled values per column. Compared with `starmie-schema`, it greatly improves actual containment while still retrieving many semantic variants. This demonstrates that semantic retrieval becomes more useful for downstream table augmentation when schema meaning is combined with value-level evidence.

Overall, value-overlap retrieval remains a strong and reliable default, schema-only retrieval is useful for discovering semantically related candidates, and weighted retrieval is the most promising direction for practical retrieve-merge-predict workflows. The next stage of this study should evaluate these retrieval choices end-to-end in the full pipeline, measuring not only which candidates are retrieved, but whether they improve downstream prediction.

References

[1] Riccardo Cappuzzo, Aimee Coelho, Felix Lefebvre, Paolo Papotti, and Gaël Varoquaux. 2025. Retrieve, Merge, Predict: Augmenting Tables with Data Lakes. *Transactions on Machine Learning Research* (2025).

[2] Shuaichen Chang, Jun Wang, Mingwen Dong, Lin Pan, Henghui Zhu, Alexander Hanbo Li, Wuwei Lan, Sheng Zhang, Jiarong Jiang, Joseph Lilien, Steve Ash, William Yang Wang, Zhiguo Wang, Vittorio Castelli, Patrick Ng, and Bing Xiang. 2023. Dr.Spider: A Diagnostic Evaluation Benchmark towards Text-to-SQL Robustness. In *International Conference on Learning Representations (ICLR)*.

[3] 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 of the 28th International Conference on Extending Database Technology (EDBT)*.

[4] Zhenwei Dai, Chuan Lei, Asterios Katsifodimos, Xiao Qin, Christos Faloutsos, and Huzefa Rangwala. 2025. LAKEGEN: A LLM-based Tabular Corpus Generator for Evaluating Dataset Discovery in Data Lakes. arXiv:2507.04687 [cs.DB]

[5] Yuhao Deng, Chengliang Chai, Lei Cao, Qin Yuan, Siyuan Chen, Yanrui Yu, Zhaoze Sun, Junyi Wang, Jiajun Li, Ziqi Cao, Kaisen Jin, Chi Zhang, Yuqing Jiang, Yuanfang Zhang, Yuping Wang, Ye Yuan, Guoren Wang, and Nan Tang. 2024. LakeBench: A Benchmark for Discovering Joinable and Unionable Tables in Data Lakes. *Proceedings of the VLDB Endowment* 17, 8 (2024), 1925–1938.

[6] Yuyang Dong, Kunihiro Takeoka, Chuan Xiao, and Masafumi Oyamada. 2021. Efficient Joinable Table Discovery in Data Lakes: A High-Dimensional Similarity-Based Approach. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. 456–467.

[7] Yuyang Dong, Chuan Xiao, Takuma Nozawa, Masafumi Enomoto, and Masafumi Oyamada. 2023. DeepJoin: Joinable Table Discovery with Pre-trained Language Models. *Proceedings of the VLDB Endowment* 16, 10 (2023), 2458–2470.

[8] 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. *Proceedings of the VLDB Endowment* 16, 7 (2023), 1726–1739.

[9] Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R. Woodward, Jinxia Xie, and Pengsheng Huang. 2021. Towards Robustness of Text-to-SQL Models against Synonym Substitution. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*.

[10] Farzaneh Mahdisoltani, Joanna Biega, and Fabian Suchanek. 2014. Yago3: A knowledge base from multilingual wikipedias. In *7th Biennial Conference on Innovative Data Systems Research*.
[11] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*. 697–706.
[12] Erkang Zhu, Fatemeh Nargesian, Ken Q Pu, and Renée J Miller. 2016. LSH Ensemble: Internet-scale domain search. In *Proceedings of the VLDB Endowment*, Vol. 9. 1185–1196.