TABERTA: A Language Model for Dataset Discovery

Enas Khwaileh
e.t.k.khwaileh@uu.nl
Utrecht University
The Netherlands

Leonard Traeger
leonard.traeger@umbc.edu
University of Maryland Baltimore County
USA

Albert Gatt
a.gatt@uu.nl
Utrecht University
The Netherlands

Yannis Velegrakis
i.velegrakis@uu.nl
Utrecht University & University of Trento
The Netherlands

Abstract

Dataset discovery requires retrieving relevant tables from large collections of relational data given a natural-language input, yet most language models are designed for unstructured text rather than structured databases. As a result, existing retrieval approaches underutilize table-specific signals such as schema organization, value distributions, and relational context. We present TABERTA, a unified retrieval framework for dataset discovery that adapts a bi-encoder language model to relational data by fine-tuning it on serialized representations of tables. The framework combines three structure-aware table serialization views, multiple retrieval-oriented fine-tuning objectives, and an offline-to-online approximate nearest-neighbor retrieval pipeline. Using WikiDBs, a large corpus of multi-table relational schemas, we fine-tune the language encoder to diverse structural patterns using optimized strategies that serialize schema and table content. Because both natural-language queries and serialized tables are encoded by the same fine-tuned model, they share a common embedding space, enabling effective semantic matching between queries and tables. Across benchmarks, TABERTA improves table retrieval by up to +60% while producing compact 768-dimensional embeddings and reducing offline indexing costs in terms of pipeline time and index size.

Keywords

Dataset Discovery, Table Retrieval, Structure-Aware Representations

1 Introduction

Dataset discovery is the process of identifying datasets relevant to a given task, typically specified through a keyword query, a natural-language description, or an example table. It is a fundamental service in data lakes and federated data management systems, where users must search over large, loosely curated collections of heterogeneous datasets. In such environments, discovery is challenging because datasets are distributed across domain-specific sources and lack consistent metadata making retrieval heavily dependent on how effectively queries are semantically matched to table representations [12, 19, 21, 30]. As data-driven practices increasingly support scientific research, business analytics, and public policy, effective dataset discovery is essential for data integration, multi-source analysis, and fact verification. However, retrieving the most relevant and complete tables remains difficult due to scale, heterogeneity, and semantic ambiguity [20, 27].

EDBT '27, Lille (France)

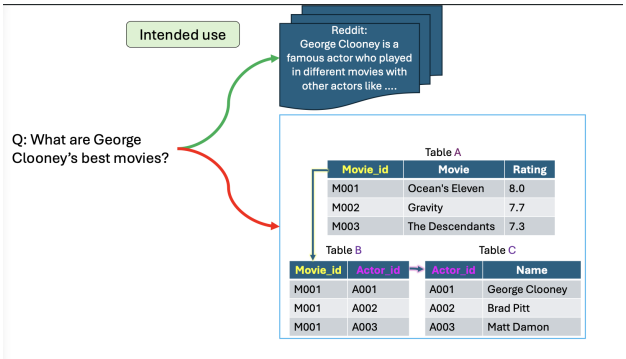

Figure 1: A Query Answering Scenario

Table Representation Learning (TRL) has emerged to address the limitations of generic text-based embeddings when applied to structured data, by encoding tables in ways that expose schema organization, column semantics, and value distributions to language models. Early TRL approaches focused primarily on single-table understanding tasks, such as entity-centric modeling, semantic parsing, or question answering (e.g., TURL [10], TaBERT [47], TAPAS [13]), or on learning dense or graph-based representations for schema and content alignment (e.g., Table2Vec [50], EmbDI [5]). While effective within their target settings, these embedding-based methods are not designed for the scalable table retrieval, such as needed in dataset discovery across large repositories. More recent work in data-lake discovery has shifted toward retrieval-oriented representation learning and fine-tuning for related-table discovery, including contrastive encoders and large-scale benchmarks for joinable or unionable table search (e.g., Starmie [3], LakeBench [11], TabSketchFM [18], DiscoverGPT [14]). These approaches demonstrate the importance of fine-tuning language models for table representations. While they focus on the task-aligned supervision for table-to-table discovery or rely on specialized pipelines with supervised signals, we aim to fine-tune an encoder model in such a way that embeddings are generically positioned more effectively in the vector space. Consequently, a fine-tuned encoder model that is aware of the inherent structure of schemas, tables, and contents will create embeddings that enable more effective and efficient nearest-neighbor retrieval of tables over large and heterogeneous data lakes.

Consider the retrieval problem illustrated in Figure 1, where we have a set of sources with documents and a query *"What are George Clooney's best movies?"*. In unstructured sources, mentions of actors, movie titles, and ratings often co-occur within a single passage, allowing text-based encoders to infer relevance

from local context. In relational settings, however, this information is found across multiple tables according to schema design: one table may capture movies and ratings, another encodes cast relationships via identifiers, and a third maps identifiers to actor names. None of these tables alone explicitly express the full intent of the query, yet each may be relevant for retrieval. While encoder-based language models are effective at matching unstructured documents with natural-language queries, their standard attention mechanism optimizes for contiguous text and falls short in encoding the relational schema structure and value organization, critically needed for retrieval.

To address this challenge, we propose TABERTA, a framework for fine-tuning a bi-directional language encoder with large database corpora so that the semantic signals of schema structure and content contextualize more precisely into more accurate table embeddings for generic data mananagement retrieval tasks. Our TABERTA framework enables efficient and effective nearest-neighbor retrieval of tables at scale across large and heterogeneous data lakes. We revisit this motivating scenario with a concrete experimental walkthrough in Section 4.3, showing how TABERTA retrieves each of the relevant tables from Figure 1.

The bi-encoder is pretrained on natural-language text and thus already captures general language semantics. Fine-tuning on serialized tables adapts the same embedding space to structured inputs expressed as text. During inference, both queries and tables are encoded by the same model into a shared vector space, enabling direct similarity-based retrieval without requiring a separate query–table alignment module.

Our contribution is a retrieval framework for dataset discovery that combines structure-aware table serialization, multiple supervision regimes studied under one retrieval setting, and an offline-to-online ANN retrieval pipeline. The training variants form a systematic design study that isolates how serialization view, supervision signal, and optimization strategy each affect retrieval quality.

In particular, We first formalize the concept of *table retrieval* as a first-class problem and define a unified retrieval setting over large relational corpora, clarifying the role of structure-aware representations in table search (Section 2). Thereafter, we propose a structure-aware table representation framework that adapts a language-model bi-encoder (SBERT) to relational data by fine-tuning it on serialized tables, enabling the schema and content within a shared embedding space for table retrieval (Section 3). In the sequel, we design and analyze multiple table serialization strategies that expose different structural views of tables, and study how these views affect representation learning and retrieval behavior (Section 3.1). Subsequently, we introduce a scalable retrieval pipeline based on offline embedding and approximate nearest-neighbor indexing, enabling efficient table search over large and heterogeneous data lakes (Section 3.3). To evaluate the proposed approach we conduct experiments using diverse table-retrieval benchmarks. The results show consistent improvements over sparse baselines and competitive performance against strong neural retrievers, with robust generalization beyond the fine-tuning corpus (Section 4). We systematically study in isolation the effects of serialization view, supervision signal, and optimization strategy on the retrieval quality across heterogeneous benchmarks (Section 4). We conclude by a discussion of the related works (Section 5).

2 Problem Statement

We consider the task of retrieving relevant tables from a corpus of relational data given a user query. Let N be an infinite set of attribute (column) names and V a set of alphanumeric values. A *schema* $S = [n_1, \ldots, n_m]$ is a sequence of m attribute names $n_i \in N$. A *tuple* is a sequence of values $[v_1, \ldots, v_m]$, where each $v_i \in V$ corresponds to attribute n_i. A *table* (relation) T is a finite set of tuples that share the same schema S. Let $\mathcal{T}$ denote the space of all possible tables (i.e., the universal table domain), and let a *corpus* $C \subseteq \mathcal{T}$ be a finite collection of tables drawn from this space.

A *query* $q \in \boldsymbol{Q}$ is a sequence of textual tokens that expresses an information need. Queries may take different forms, including keyword queries (e.g., "Clooney movies"), natural-language questions (e.g., "What are George Clooney's best movies?"), or factual statements requiring verification (e.g., "George Clooney acted in Ocean's Eleven").

The goal of *table retrieval* is to rank tables in a corpus according to their relevance to a query. Formally, given a query $q \in \boldsymbol{Q}$ and a corpus C, a retrieval model returns a ranked list $\hat{R}_q = [T_1, \ldots, T_k]$, where $T_i \in C$, satisfying

$$\forall\, T_i \in \hat{R}_q,\ \forall\, T_j \in C \setminus \hat{R}_q, \quad \mathrm{rel}(q, T_i) \geq \mathrm{rel}(q, T_j),$$

where $\mathrm{rel}(q, T)$ denotes the relevance of table T to query q. The tables in $\hat{R}_q$ are ordered by non-increasing relevance:

$$\mathrm{rel}(q, T_1) \geq \mathrm{rel}(q, T_2) \geq \cdots \geq \mathrm{rel}(q, T_k).$$

We use $\hat{R}_q$ to refer to the retrieved ranking throughout; the retrieval pipeline (Sections 3.3–3.4) instantiates this set via ANN search over TABERTA table embeddings. In this work, we focus on learning structure-aware representations for queries and tables that enable effective matching using standard similarity measures. For tasks that require textual outputs rather than ranked tables (e.g., question answering), an optional decoding component can be applied after retrieval. We define a decoder

$$G_{\mathrm{dec}} : (\boldsymbol{Q}, \mathcal{T}^k) \to \mathcal{A},$$

which generates a natural-language answer $a \in \mathcal{A}$ conditioned on the query q and the retrieved tables $\hat{R}_q$. The decoder is an external, independently trained sequence-to-sequence model that receives the query and the top-k retrieved tables as input. Retrieved tables are converted into compact evidence snippets consisting of the table name, schema (column names), and up to five representative rows, concatenated under a fixed token budget. No joint encoder-decoder training is performed: the fine-tuned TABERTA encoder provides ranked evidence, and the decoder generates answers independently. Retrieval remains the primary task, and answer generation is treated as a downstream application that depends on the quality of retrieved tables.

3 Learning Structure-Aware Table Representations

We study how language models can be adapted to represent and retrieve relational tables in response to natural-language queries. Because relational evidence may be localized within a table or distributed across multiple tables in a database, table serialization is treated as a design dimension: different views expose different structural and semantic cues to a bi-encoder. Concretely, tables are represented as text sequences that preserve selected tabular signals, and the encoder is fine-tuned with retrieval-oriented objectives. Figure 2 summarizes the workflow, distinguishing

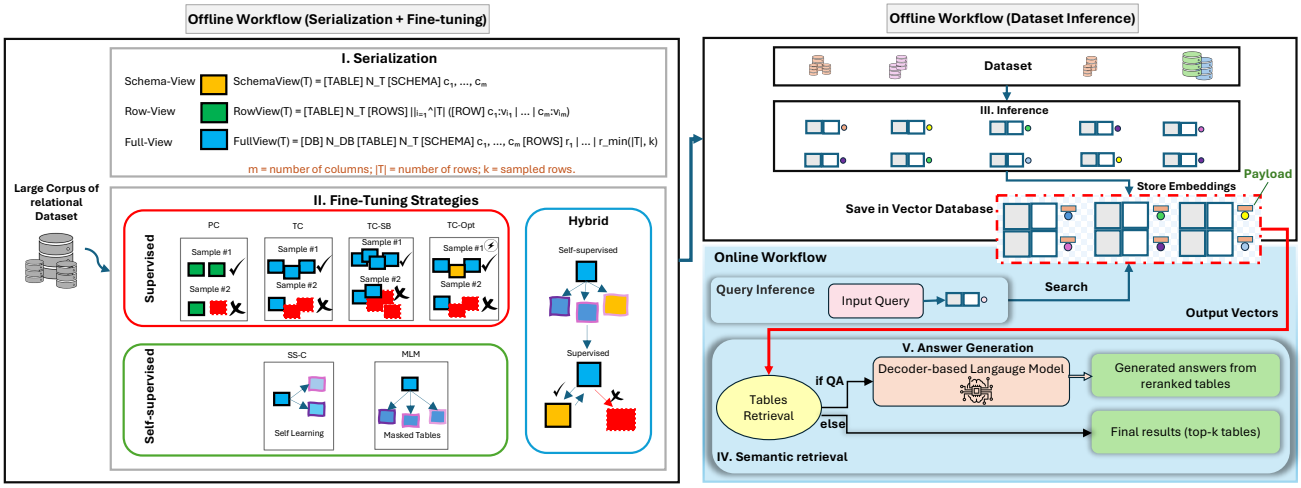

Figure 2: The TABERTA framework

offline representation learning and indexing (Steps I–III) from online retrieval and optional answer generation (Steps IV–V).

The framework consists of three reusable components shared across all variants: (1) structure-aware table serialization into text, (2) bi-encoder fine-tuning on a large relational corpus, and (3) offline embedding with approximate nearest-neighbor indexing for scalable retrieval. Within this framework, we study seven training configurations that differ along three controlled axes: serialization view (SchemaView, RowView, FullView), supervision signal (supervised, self-supervised, hybrid), and optimization strategy (standard, smart batching, AMP + checkpointing). These variants form a systematic design study whose goal is to identify which factors most affect retrieval quality across heterogeneous benchmarks.

3.1 Table Representation and Serialization Views

Given a table T, serialization specifies how its schema and content are linearized into a text sequence consumed by the encoder. We formalize this process through the serialization function that differ in the amount and type of information exposed to the model.

Serialization function. Let $\mathcal{T}$ denote the space of all tables, and let $\mathbb{X}$ be the space of serialized strings. We define a serialization function $\text{serialize}_\theta : \mathcal{T} \to \mathbb{X}$, where $\theta \in \{\textsc{SchemaView}, \textsc{RowView}, \textsc{FullView}\}$ selects a particular strategy that emphasizes different structural and semantic aspects of a table. For a table T, we write N_{DB} for its database identifier, N_T for the table name, $c_1, \ldots, c_m$ for column headers, and v_{ij} for the value in row i, column j. We denote the i-th row by the ordered sequence $r_i = [v_{i1}, \ldots, v_{im}]$. We use bracketed delimiter markers such as [DB], [TABLE], [SCHEMA], [ROW], and [ROWS] to make the role of each serialized segment explicit. For TABERTA, the functional role of these markers is learned during fine-tuning; for frozen baselines, the same markers are treated only as textual delimiters under the fixed serialization protocol.

Schema-only view. This view encodes the table identity and column headers while omitting all cell values:

$$\textsc{SchemaView}(T) = \text{[TABLE]}\ N_T\ \text{[SCHEMA]}\ c_1, \ldots, c_m\ .$$

SchemaView emphasizes structural signals such as attribute names and schema layout. It is lightweight and robust to table size, and is particularly effective when relevance can be inferred from column semantics alone (e.g., entity-focused or keyword queries).

Row-level view. This view represents table content as a sequence of row strings, where each row is expressed as column-value pairs:

$$\textsc{RowView}(T) = \text{[TABLE]}\ N_T\ \text{[ROWS]}\ \Big\|_{i=1}^{|T|} \Big(\text{[ROW]}\ c_1{:}v_{i1} \mid \cdots \mid c_m{:}v_{im}\Big).$$

Here, $|T|$ denotes the number of rows in table T, $\|$ denotes string concatenation, with row strings separated by a special delimiter token (e.g., [SEP]), and $c_1, \ldots, c_m$ are the m column headers from the schema S. RowView exposes fine-grained value-level evidence while preserving column context. This view is suited for retrieval scenarios where relevance depends on specific cell values (e.g., facts, numeric attributes, or entity mentions), but scales linearly with the number of rows.

Full structured view. This view concatenates lightweight database context, schema information, and a controlled amount of table content into a single sequence:

$$\textsc{FullView}(T) = \text{[DB]}\ N_{DB}\ \text{[TABLE]}\ N_T\ \text{[SCHEMA]}\ c_1, \ldots, c_m\ \text{[ROWS]}\ \Big\|_{i=1}^{|T|} r_i.$$

FullView provides the richest representation, combining schema cues with broader content context. It allows the encoder to jointly model how attributes and values co-occur within a table, at the cost of higher input length.

Input length and sampling. To respect encoder length limits, the full schema is retained and only a fixed number of rows (or a deterministic row sample) is included for large tables. Specifically, for tables whose serialized representation exceeds the encoder's input limit (512 tokens), we retain the complete schema and include up to k=5 rows using a first-k deterministic sampling policy. This preserves schema-level semantics and representative content: the resulting embedding captures the table's structural and semantic profile rather than memorizing individual rows. When fine-grained cell-level matching is critical (e.g.,

evidence-heavy retrieval), RowView-based row-level indexing (Section 3.3) can complement table-level embeddings by indexing each row independently. FullView provides the most comprehensive serialization within the framework by combining database context, schema information, and sampled row evidence, whereas SchemaView offers a compact alternative for settings where schema-level cues are sufficient. Together with RowView, these serialization views make TABERTA configurable across retrieval scenarios, allowing practitioners to select the amount of table evidence encoded during offline indexing without changing the online retrieval interface. The views are paired with different fine-tuning objectives in Section 3.2.

3.2 Fine-Tuning Strategies

During the offline phase (Step II in Figure 2), the encoder $E(\cdot)$ is fine-tuned on serialized tables to produce embeddings suitable for similarity-based retrieval. The training strategies below are used as controlled fine-tuning variants within a common TABERTA retrieval framework, rather than as independent model contributions. All strategies target the same downstream objective: ranking tables by relevance to a natural-language query using embedding similarity. They differ along three controlled dimensions: learning type (supervised, self-supervised, or hybrid), learning objective (contrastive, triplet, or masking), and serialization view (SchemaView, RowView, FullView, or a combination), as summarized in Table 1.

This separation clarifies the contribution scope. The methodological contribution lies in the structure-aware serialization views, the construction of relationally grounded positive and negative training signals, and the staged Hybrid pipeline that first adapts the encoder to tabular context through MLM and then aligns the embedding space through triplet-based retrieval supervision. The underlying losses, including InfoNCE, margin-based triplet loss, and MLM, are adopted objectives used to instantiate and compare these variants under a unified retrieval protocol. Fine-tuning is corpus-agnostic and performed once on a large collection of relational databases, and the resulting encoder is reused unchanged during the online retrieval phase.

Formally, for a table T and a serialization view θ, we denote by $x_T^\theta = \textsc{serialize}_\theta(T)$ its serialized textual representation and by $\mathbf{h}_T^\theta = E(x_T^\theta) \in \mathbb{R}^d$ the corresponding encoder embedding. Cosine similarity between two embeddings $\mathbf{h}_i$ and $\mathbf{h}_j$ is written as $\cos(\mathbf{h}_i, \mathbf{h}_j)$.

Supervised Approaches. Supervised fine-tuning adapts the encoder $E(\cdot)$ using automatically derived table-level supervision from relational databases. Given a serialized table T_i, we obtain its representation $\mathbf{z}_i = E(T_i)$. Positive table pairs (T_i, T_j) are constructed based on relational context, including co-occurrence within the same database, shared schema elements, or explicit foreign-key links, while unrelated tables are treated as negatives.

Pairwise Contrastive Training (PC). This strategy formulates supervised table retrieval as a pairwise contrastive learning problem using the InfoNCE objective [44]. Given an anchor table T_a, the encoder is trained to bring a related table T_p closer in the embedding space while pushing unrelated tables farther away. Positive pairs are constructed automatically from relational context, such as co-occurrence within the same database or connections via schema metadata. All other tables in the same mini-batch act as negatives, and when available, mined hard negatives $\mathcal{N}_a$ are included to strengthen the contrastive signal.

For this objective, each table is serialized using the RowView strategy, which emphasizes fine-grained column–value associations at the row level. The encoder produces representations $\mathbf{z}_a = E(T_a)$ and $\mathbf{z}_p = E(T_p)$. The training loss is defined as:

$$\mathcal{L}_{\text{pair}}(T_a) = -\log \frac{\exp\big(\cos(\mathbf{z}_a, \mathbf{z}_p)/\tau\big)}{\sum_{j \in \mathcal{B}} \exp\big(\cos(\mathbf{z}_a, \mathbf{z}_j)/\tau\big)}, \tag{1}$$

where $\cos(\cdot,\cdot)$ denotes cosine similarity, τ is a temperature parameter, and $\mathcal{B}$ denotes the mini-batch.

We adopt RowView for pairwise contrastive training because it highlights localized overlap patterns between related tables through aligned column-value pairs.

Triplet-Based Ranking Training (TC). This strategy extends supervised contrastive learning by explicitly modeling relative relevance through ordered table triplets. Rather than only separating related from unrelated tables, the objective enforces that a related table should be closer to the anchor than an unrelated one by a fixed margin, directly reflecting the ranking nature of table retrieval.

Training samples are constructed as triplets (T_a, T_p, T_n), where T_a is an anchor table, T_p is a semantically or structurally related table derived from relational context, and T_n is an unrelated table. Each table is serialized using the FullView strategy, and the encoder produces representations $\mathbf{z}_a = E(T_a)$, $\mathbf{z}_p = E(T_p)$, and $\mathbf{z}_n = E(T_n)$. The model is trained using a margin-based triplet loss:

$$\mathcal{L}_{\text{triplet}}(T_a, T_p, T_n) = \max\Big\{0,\ \cos(\mathbf{z}_a, \mathbf{z}_n) - \cos(\mathbf{z}_a, \mathbf{z}_p) + \alpha\Big\}, \tag{2}$$

where α is a margin hyperparameter. We adopt FullView for this objective because relative relevance comparisons require holistic table representations that integrate schema structure and global content, enabling reliable discrimination between competing candidates in a ranking setting.

TC-Opt and TC-SB. Both variants preserve $\mathcal{L}_{\text{triplet}}$ unchanged and differ only in engineering choices. **TC-Opt** additionally exposes the encoder to SchemaView alongside FullView, and enables automatic mixed precision [28] with gradient checkpointing [8] for longer inputs and larger effective batches. **TC-SB** applies smart batching [34], grouping semantically related tables per mini-batch to produce more competitive ranking comparisons without altering the loss.

While supervised fine-tuning derives training signals from externally defined relational context such as table co-occurrence or schema links this supervision can be sparse or unavailable in heterogeneous data lakes. To relax this dependency, we adopt self-supervised learning, where supervision is induced directly from individual tables by contrasting multiple stochastic views of the same serialized input, enabling the encoder to learn invariant representations without relying on database-level relations.

Self-Supervised Approaches.

Self-Supervised Contrastive (SimCSE). Learns table embeddings without explicit relational supervision by enforcing consistency across complementary views of the same table. Given a table T, we serialize it using SchemaView(T) and FullView(T), then independently encode each under dropout to obtain stochastic embeddings $\mathbf{z}^{(1)}$ and $\mathbf{z}^{(2)}$ treated as a positive pair. The same InfoNCE objective as $\mathcal{L}_{\text{pair}}$ is applied with all other batch tables as implicit negatives. Combining SchemaView and FullView

Table 1: Fine-tuning Strategies

Strategy	Learning	Training Objective	Serialization View(s)	Granularity
Pairwise Contrastive *(PC)*	Supervised	InfoNCE (pairwise contrastive)	RowView	Row
Triplet Contrastive *(TC)*	Supervised	Margin-based triplet loss (TML)	FullView	Table
Triplet Contrastive (SmartBatch) *(TC-SB)*	Supervised	TML(+ smart batching)	FullView	Table
Triplet Contrastive (Optimized) *(TC-opt)*	Supervised	TML(+ AMP, checkpointing)	FullView + SchemaView	Table
Self-Supervised Contrastive (SimCSE) *(SS-C)*	Self-superv.	Contrastive (in-batch negatives)	SchemaView + FullView	Table
Masked Language Modeling *(MLM)*	Self-superv.	MLM loss	FullView (masked)	Token
Hybrid *(MLM + TC)*	Hybrid	MLM → TML (staged)	SchemaView → FullView	Table

encourages view-invariant representations that jointly capture schema structure and table content.

Masked Language Modeling (MLM). MLM adapts the encoder by learning to recover missing tokens from the surrounding table context, providing a self-contained training signal that does not rely on relational supervision. Given a serialized table T, we randomly mask a subset $\mathcal{M}$ of its tokens and train the encoder to predict each masked token conditioned on the unmasked context.

$$\mathcal{L}_{\text{MLM}} = -\sum_{i \in \mathcal{M}} \log P(w_i \mid \{w_j\}_{j \notin \mathcal{M}}), \tag{3}$$

where $\mathcal{M}$ denotes the set of masked positions. We apply MLM using the FullView serialization, which exposes the encoder to complete schema information and global table content, allowing masked tokens to be inferred from both structural cues and contextual dependencies. The resulting encoder is then used to compute table representations $\mathbf{z} = E(T)$ for downstream retrieval, ensuring that token and schema-level contextualization directly shapes the embeddings used by subsequent retrieval objectives.

Hybrid (MLM + TC). The hybrid strategy combines masked language modeling with triplet-based retrieval fine-tuning. Its purpose is to separate table-aware representation adaptation from retrieval-specific alignment. In the first stage, the encoder is adapted to tabular structure using MLM over FullView serializations:

$$\phi^{(1)} = \arg\min_{\phi} \; \mathbb{E}_T\Big[\mathcal{L}_{\text{MLM}}\big(x_T^{\text{FullView}}; \phi\big)\Big],$$

where $x_T^{\text{FullView}} = \text{serialize}_{\text{FullView}}(T)$ denotes the serialized representation of table T. This stage provides a self-supervised signal that does not require positive or negative table pairs. Instead, it exposes the encoder to table-internal dependencies, including associations between column names, cell values, table identifiers, and database context. As a result, the pretrained text encoder is first adapted from sentence-level semantics to tabular regularities before any retrieval supervision is introduced.

In the second stage, the MLM-adapted encoder is further optimized for retrieval using triplet supervision. Starting from $\phi^{(1)}$, we fine-tune the encoder on relationally derived triplets (T_a, T_p, T_n), where T_a is an anchor table, T_p is a related table, and T_n is an unrelated table:

$$\phi^{(2)} = \arg\min_{\phi} \; \mathbb{E}_{(T_a,T_p,T_n)}\Big[\mathcal{L}_{\text{triplet}}\big(T_a, T_p, T_n; \phi\big)\Big].$$

During this stage, we use multi-view serialization with SchemaView and FullView, so that the retrieval signal is informed by both compact schema-level cues and value-level table content. This is important for dataset discovery because relevance may depend on column semantics, factual values, or the structural role of a table within a relational database. The staged design therefore avoids forcing the model to learn retrieval ordering from under-adapted table embeddings: MLM first stabilizes table-aware contextual representations, and triplet fine-tuning then reshapes this adapted space for nearest-neighbor table retrieval.

3.3 Embedding Construction and Indexing

After offline fine-tuning, the encoder $E^*(\cdot)$ is fixed and used to materialize vector representations for all tables in the target repository. This step operationalizes the learned representations by transforming serialized tables into dense embeddings that can be stored, indexed, and reused during online retrieval.

Let $T \in \mathcal{T}$ be a table and let Θ denote the serialization view(s) associated with the fine-tuning strategy. For single-view strategies, $\Theta = \{\theta\}$ with $\theta \in \{\text{SchemaView}, \text{RowView}, \text{FullView}\}$; for multi-view strategies, $\Theta = \{\theta_1, \theta_2\}$ (e.g., $\{\text{SchemaView}, \text{FullView}\}$), denote by:

$$x_T^{\theta} = \text{serialize}_{\theta}(T), \qquad \mathbf{h}_T^{\theta} = E^*(x_T^{\theta}) \in \mathbb{R}^d.$$

When multiple views are used, we construct a single table embedding by view aggregation,

$$\mathbf{v}_T = \text{Agg}\big(\{\mathbf{h}_T^{\theta}\}_{\theta \in \Theta}\big),$$

where $\text{Agg}(\cdot)$ is a fixed aggregation operator (e.g., mean pooling), yielding a compact embedding compatible with nearest-neighbor retrieval.

The embedding granularity depends on the fine-tuning strategy. For row-level training (Pairwise Contrastive with RowView), we embed each row r_i independently,

$$\mathbf{v}_{r_i} = E^*\big(\text{serialize}_{\text{RowView}}(r_i)\big),$$

and associate it with key (T, i). For table-level strategies (Triplet, SimCSE, MLM, and Hybrid), we compute a single holistic embedding $\mathbf{v}_T$ per table. Supporting both granularities enables fine-grained matching when local evidence is informative and holistic retrieval when global semantics dominate.

Lightweight metadata (table name, schema, foreign keys, and database identifier) are stored alongside each embedding to support downstream reranking and result presentation. For row-level strategies (PC with RowView), one embedding per row is indexed and table-level scores are derived by aggregation; for all other strategies, a single table-level embedding $\mathbf{v}_T$ is indexed directly. All embeddings are inserted into an HNSW approximate nearest-neighbor index [26] due to its favorable recall–latency tradeoff.

3.4 Semantic Retrieval

At query time, the query q is encoded by the same fine-tuned encoder without table-specific serialization, $\vec{v}_q = E^*(q)$, placing it in the same vector space as all pre-computed table embeddings. For table-level strategies, candidate tables are ranked by $\cos(\vec{v}_q, \vec{v}_T)$; for row-level strategies (PC), per-row cosine scores are aggregated by max pooling to produce a table-level score. The top-k tables are returned as the ranked result $\hat{R}_q$. Optionally, a

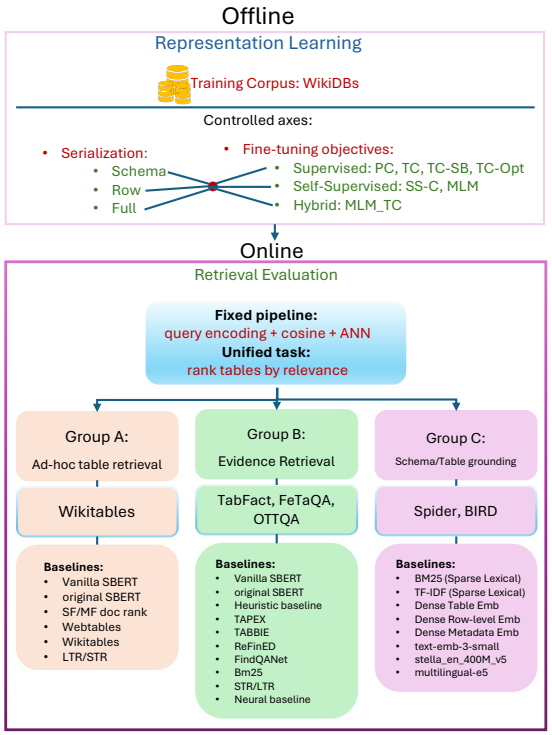

Figure 3: TABERTA Evaluation Overview.

lightweight schema-overlap bonus $s_{\text{col}}(q, T) = |Q \cap S_T|/|Q \cup S_T|$ (with small weight $\lambda \in [0, 0.3]$) breaks near ties without altering the primary cosine ranking.

Downstream answer generation. Some evaluation settings require producing a natural-language answer in addition to returning ranked tables. This step is optional and is applied only after retrieval. TABERTA first ranks tables using the encoder and cosine-similarity pipeline described above; the resulting embeddings are not passed to the generator. Instead, the top-k retrieved tables are converted into textual evidence snippets and concatenated with the query under a fixed token budget. Each snippet contains lightweight table metadata, the schema, and up to p representative rows. For table-level indexing, these rows are selected from the serialized table sample; for row-level indexing, they correspond to the highest-scoring retrieved rows. The packed textual input can be consumed by any sequence-to-sequence decoder, such as BART or T5, which then generates the final answer. Because the decoder operates only on packed evidence, changing the decoder architecture does not alter TABERTA's embedding construction, ANN indexing, similarity scoring, or table ranking.

4 Experimental Evaluation

We evaluate whether structure-aware fine-tuning of a bi-encoder produces table embeddings that are both effective and transferable across heterogeneous table-retrieval settings. Figure 3 summarizes the overall evaluation design.

Our evaluation decouples representation learning from downstream benchmarking. In the offline phase, we fine-tune encoders on WikiDBs [45] while controlling two factors: the serialization view used to expose table structure (schema, row, or full view) and the fine-tuning objective corresponding to the training strategies. In the online phase, all models are evaluated under a unified table-retrieval formulation with a fixed retrieval pipeline.

Under this controlled setup, we examine three questions: (i) whether structure-aware fine-tuning improves retrieval quality over sparse and generic dense baselines; (ii) how different serialization views affect retrieval behavior; and (iii) whether a single fine-tuned encoder generalizes across diverse downstream settings, including ad-hoc tablesearch, evidence retrieval, and schema/table grounding. Each component of the evaluation design in Figure 3 is detailed in the following subsections.

4.1 Datasets and Benchmarks

We distinguish the corpus used for representation learning from downstream benchmarks used only for evaluation.

4.1.1 Fine-tuning. We fine-tune all models on WikiDBs, a large corpus of relational databases automatically extracted from Wikidata, it contains over 100K databases and 1.6M tables spanning diverse domains and realistic schema designs. WikiDBs is used exclusively for table-level representation learning, no benchmark-specific queries, relevance labels, or task annotations are observed during fine-tuning.

4.1.2 Retrieval Benchmarking Tasks. By organizing benchmarks along these dimensions, we enable a consistent comparison of serialization views and fine-tuning objectives across heterogeneous retrieval settings. Table 2 summarizes the experimental datasets.

Group A: Ad-hoc Table Retrieval. This group evaluates retrieval scenarios where relevance is primarily determined by schema semantics rather than table content. We use the **WikiTables** benchmark [51], which focuses on keyword-based retrieval of web tables using short queries (one to five tokens). In this setting, relevance is driven by column names, table titles, and metadata, making it a direct test of schema-level matching. As such, this group assesses whether schema-focused table representations are sufficient for effective retrieval without requiring access to detailed cell values.

Group B: Evidence Table Retrieval (Value-grounded relevance). This group includes benchmarks where retrieval relevance depends on identifying tables that contain specific factual values or multi-row evidence rather than matching schema descriptions alone. We evaluate on **TabFact** [6], a fact verification benchmark in which tables must provide evidence to support or refute a natural-language claim; **FeTaQA** [29], a free-form question answering dataset requiring explanatory answers derived from multiple rows; and **OTTQA** [7], an open-domain table QA benchmark with short factoid queries and partially implicit table annotations. Together, these datasets stress the need for content-aware table representations capable of grounding queries in factual table values.

Group C: Schema and Table Grounding (Structural relevance). This group evaluates retrieval settings where relevance is defined by correct grounding to database schemas or tables that support structured query interpretation. We use the **Spider** [48] and **BIRD** [49] benchmarks, where gold-relevant tables are derived from SQL queries. Although these datasets are used for text-to-SQL generation, we restrict evaluation to the retrieval task of identifying the correct schemas or tables.

Table 2: Dataset Overview

Dataset	Benchmark Group	Retrieval Semantics	#Tables	#Queries
WikiTables	Ad-hoc Table Retrieval	Schema-driven relevance	1.6M	60
TabFact	Evidence Table Retrieval	Value-grounded evidence	16.6K	25.6K
FeTaQA	Evidence Table Retrieval	Multi-row factual evidence	3K	2K
OTTQA	Evidence Table Retrieval	Factoid evidence grounding	789	2.2K
Spider	Schema/Table Grounding	Schema identification	1K	2.1K
BIRD	Schema/Table Grounding	Schema identification	75	1.5K

4.1.3 Baseline Methods. Baseline methods are selected and reported according to the retrieval semantics of each benchmark groups (A, B, and C). All baselines are evaluated under the same retrieval protocol as TABERTA, using identical query inputs, similarity functions, indexing mechanisms, and evaluation metrics. All methods are used strictly as *retrievers*; we do not compare end-to-end execution or generation systems whose performance depends on downstream reasoning or query execution.

Group A: Ad-hoc Table Retrieval. For schema-driven ad-hoc table retrieval on WikiTables, we compare against baselines commonly used for keyword-based table search, where relevance is primarily determined by schema semantics, including column names and table titles. These include sparse lexical retrievers such as BM25 [36] and TF–IDF [37], which provide competitive reference points for assessing retrieval effectiveness in schema-focused settings.

We additionally report classical table-search and learning-to-rank systems, including WebTable and single-field (SF) / multi-field (MF) document ranking variants [4, 31], WikiTable [2], and the LTR and STR models of Zhang et al. [51]. These baselines contextualize NDCG performance under schema-focused retrieval without learned table embeddings. Finally, we include generic dense encoders such as Vanilla SBERT [35] and Original SBERT [40] to assess the benefit of structure-aware fine-tuning over untuned text representations (Table 5).

Group B: Evidence Table Retrieval. For evidence-driven retrieval on TabFact, FeTaQA, and OTTQA, relevance depends on identifying tables that contain specific factual values or multi-row evidence supporting a query or claim. In this setting, purely schema-based or lexical methods are insufficient. We therefore compare against sparse lexical baselines (BM25), generic dense encoders (Vanilla SBERT [35], Original SBERT [40]), and structure-aware neural table encoders including TAPAS [13], TAPEX [25], and TABBIE [15]. Although originally designed for downstream reasoning tasks, these models provide strong references for content-aware table representations when used as retrievers. For TabFact, we additionally report task-specific reference systems for factual grounding over tables, including ReFinED [1] and FinQANet [9], as well as a simple heuristic baseline. These baselines establish competitive upper bounds for value-grounded retrieval despite being optimized for downstream verification rather than retrieval (Tables 6, 7).

Group C: Schema and Table Grounding. For schema and table grounding on Spider and BIRD, retrieval relevance is defined by whether the correct schemas or tables required for structured query interpretation are retrieved. We compare against sparse lexical retrievers (BM25 and TF–IDF), which capture surface-level overlap between queries and schema elements, as well as a broad set of dense retrieval baselines evaluated under the TARGET framework [17]. These include table-level, row-level, and metadata-level dense embeddings, along with representative off-the-shelf embedding models such as text-embedding-3-small, multilingual-e5-large-instruct, and stella_en_400M_v5. We also include untuned sentence encoders (Vanilla SBERT [35], Original SBERT [40]) to quantify the gap between generic dense representations and structure-aware fine-tuning. These baselines reflect the state of practice for schema grounding and provide a strong reference for evaluating retrieval robustness across heterogeneous database schemas (Table 8). We evaluate Qwen3-Embedding-8B [42] as a frozen decoder-only embedding baseline under the same retrieval protocol, results are consolidated in Table 4.

Table 3: Fine Tuning Training Strategy Details

Strategy	Samples	Best Ep.	Best Loss	Time	E.Stop
PC‡	89,864	2	0.005	121h 40m	Yes
SS-C (SimCSE)	38,224	9	0.000	40h 40m	Yes
TC	9,000	7	3.665	13h 53m	Yes
TC-Opt	9,000	5	3.866	14h 00m	Yes
TC-SB	9,000	5	3.655	14h 00m	Yes
MLM	38,225	49	0.365	32h 08m	No
Hybrid (MLM+TC)	9,000	7†	3.868	$47h\,58m^{*}$	Yes

Shared: lr=2 × 10^{-5}, batch=16, max-seq=512, seed=42, max epochs=200

4.2 Experimental Setup

Models and Fine-Tuning Strategies. We fine-tune and evaluate seven SBERT variants (`all-mpnet-base-v2`) using the WikiDBs corpus [45] for training and evaluation on various downstream benchmarks. All evaluations are conducted under a unified table retrieval setting, where a natural-language or keyword query is used to rank candidate tables.

Table 3 reports per-strategy training details under a maximum budget of 200 epochs with early stopping (patience = 5 on validation loss). The base encoder is `all-mpnet-base-v2` (109M parameters, max_seq_len = 384) for all strategies except MLM, which uses `microsoft/mpnet-base` (max_seq_len = 512). Shared settings: batch = 16, lr = 2×10^{-5} (AdamW), seed = 42, τ=0.05, α=0.5 (triplet margin), first-k row sampling with k=5. All strategies converge well within the 200-epoch budget; the latest best epoch is 49 (MLM). Val losses are not directly comparable across strategies as they use different loss scales.

Retrieval and Indexing. The evaluation pipeline follows the TARGET benchmark framework [17], extended with dense retrieval implemented using Qdrant [33] for scalable vector search. Query and table embeddings are indexed with cosine similarity and approximate nearest neighbor search via HNSW [26], with optional structure-aware re-ranking applied after initial retrieval. All experiments are conducted on a single machine equipped with an `NVIDIA A100-40GB GPU`, an `Intel Xeon` CPU, and `256 GB` of memory.

Evaluation Metrics. We report standard retrieval metrics commonly used in information retrieval and table search. Each metric captures a different aspect of retrieval quality: *MAP* (Mean Average Precision) and *MRR* (Mean Reciprocal Rank) emphasize the rank of the first relevant result, measuring early/exact precision, a model scores zero on these metrics if no relevant table appears at rank 1 across all queries. *NDCG@k* (Normalized Discounted Cumulative Gain) captures graded ranking quality across

the full top-k list, rewarding models that place relevant items higher even if not at rank 1. *CR@k* (Containment Recall) measures the fraction of gold-relevant tables appearing anywhere in the top-k, assessing coverage. *Precision*, *Recall*, and *F1* measure set-based overlap between retrieved and gold tables. *BLEU* (SacreBLEU [32]) measures downstream answer quality, applied only to QA benchmarks where an external decoder generates answers from retrieved evidence. This is why a model can achieve MAP=0 yet high NDCG: it may fail to rank a relevant table first while still placing relevant items within the broader top-k cutoff. WikiTables is evaluated using MAP, MRR, and NDCG@5/10/20/60 [16]. Spider and BIRD use containment recall (CR@1/5/10), MRR@10, and NDCG@10 to assess schema and table grounding quality. FeTaQA and OTTQA are evaluated using Recall@k for table retrieval, together with SacreBLEU [32] for answer generation quality. TabFact reports Precision@k, Recall@k, and F1@k over evidence-table retrieval, following prior work [6].

Efficiency and Resource Usage. Table 4 reports empirically measured offline embedding and indexing efficiency for all TABERTA variants and Qwen3-Embedding-8B across the five evaluated corpora, covering 64,263 tables on an NVIDIA A100-PCIE-40GB GPU. All variants produce 768-dimensional embeddings, whereas Qwen3-Embedding-8B produces 4096-dimensional embeddings.

Across TABERTA variants, the total offline embedding and indexing pipeline requires 1,578–1,999 s. Hybrid requires 1,831 s in total, compared with 4,684 s for Qwen3-Embedding-8B, corresponding to a 60.9% reduction in offline pipeline time. Considering encoding alone, Hybrid requires 1,515 s compared with 4,155 s for Qwen3-Embedding-8B. Storage is also lower: TABERTA embeddings require 2.93 MB per 1,000 embeddings, compared with 15.63 MB per 1,000 embeddings for Qwen3-Embedding-8B. At WikiTables scale, this corresponds to approximately 4.7 GB for TABERTA variants versus 25 GB for Qwen3-Embedding-8B.

4.3 Results by Benchmark Group

Results are reported by benchmark group to reflect different retrieval semantics: schema-driven ad-hoc search (Group A), value-grounded evidence retrieval (Group B), and schema/table grounding derived from structured queries (Group C). All models follow the same retrieval protocol, so differences reflect the interaction between benchmark semantics and the controlled modeling choices (serialization view and fine-tuning objective). This pattern is consistent with objective bias: strongly separating positives improves early precision, while triplet supervision improves ranking depth.

4.3.1 Group A: Ad-hoc Table Retrieval. We first evaluate ad-hoc table retrieval on the WikiTables benchmark [51], which represents schema-driven retrieval with short keyword queries. In this setting, relevance is largely determined by schema-level metadata, including table titles and column names, making it a direct test of schema-level matching rather than value grounding.

Table 5 reports retrieval performance across our fine-tuning strategies and prior baselines, measured using MAP, MRR, and NDCG at multiple cutoffs. Structure-aware fine-tuning consistently improves over generic sentence encoders, confirming that exposing relational schema information benefits ad-hoc table retrieval.

Among our models, Hybrid (MLM+TC) achieves perfect MAP and MRR, indicating highly accurate identification of the top-ranked relevant table. However, its NDCG scores are lower than those of triplet-based variants, suggesting that while Hybrid excels at early precision, it produces less stable rankings at deeper cutoff levels. In contrast, TC and especially TC-opt achieve the strongest NDCG@10/20/60, highlighting the effectiveness of triplet-based supervision for producing consistent ranking distributions.

This distinction reflects the different optimization biases of the objectives. Objectives that strongly separate positive and negative examples (e.g., PC and Hybrid) favor exact top-rank accuracy, whereas triplet-based objectives explicitly optimize relative ordering among multiple candidates, leading to improved ranking depth. Figure 4 illustrates this divergence by contrasting exact-match metrics (MAP, MRR) with ranking-based metrics (NDCG).

Classical table-search baselines such as STR and LTR [51] remain competitive on NDCG, underscoring the importance of explicit schema term matching for keyword-driven retrieval. Nevertheless, several structure-aware fine-tuned models match or surpass these baselines, demonstrating that learned table representations can capture schema semantics beyond lexical overlap. Sparse lexical baselines and untuned sentence encoders perform substantially worse, particularly on early precision metrics.

4.3.2 Group B: Evidence-Centric Table Retrieval.

Fact Verification (TabFact). The TabFact dataset [6] evaluates fact verification over tables, where each natural-language claim must be supported or refuted using factual evidence contained in a table. Unlike ad-hoc retrieval, relevance in this setting is *value-grounded*: successful retrieval requires aligning claims with specific cell values or multi-row evidence rather than relying on schema descriptions alone. Accordingly, retrieval quality is measured using Precision, Recall, and F1 score.

Given this value-dependent notion of relevance, we evaluate TabFact using *content-aware serializations* (FullView and, where applicable, RowView), rather than schema-only views, to ensure access to factual evidence during retrieval. Table 6 reports results for our fine-tuning strategies alongside reference baselines.

Among our models, the Hybrid (MLM+TC) strategy achieves the highest recall (0.9945) and the highest F1 score (0.7053), indicating that it successfully retrieves nearly all evidence-bearing tables. This behavior reflects the benefit of combining content-aware representation learning through masked language modeling with supervised ranking objectives that explicitly optimize retrieval quality.

Triplet-based strategies (TC, TC-opt, and TC-SB) achieve a strong balance between precision and recall, with F1 scores above 0.60, showing that supervised contrastive objectives are effective for preserving factual consistency when table content is exposed. In contrast, the self-supervised contrastive baseline (SS-C) performs poorly, highlighting the difficulty of learning value-level grounding without explicit supervision. Generic sentence encoders (Vanilla SBERT, Original SBERT) lag behind, confirming that untuned text representations are insufficient for evidence-driven table retrieval.

We compare against specialized table reasoning and fact verification models, including TAPAS [13], TAPEX [25], TABBIE [15], ReFinED [1], and FinQANet [9]. Although these systems were designed for downstream reasoning, the Hybrid retriever achieves competitive F1 performance, narrowing the gap to task-specific architectures despite operating purely at the retrieval level.

While BM25 and table-search rankers (LTR, STR) [51] perform competitively by exploiting schema and metadata cues, they lack

Table 4: TABERTA variants and Qwen3-Embedding-8B Efficiency

Model	Dim	Enc. (s)	Idx. (s)	Total (s)	Tput (e/s)	ms/emb	Emb. (MB)	Qdrant (MB)	/1K (MB)
PC	768	1620.64	38.15	1999.29	32.14	31.11	188.27	356.36	2.93
SS-C (SimCSE)	768	1608.86	37.65	1955.68	32.86	30.43	188.27	356.52	2.93
TC	768	1239.14	37.55	1578.47	40.71	24.56	188.27	356.41	2.93
TC-Opt	768	1502.85	37.89	1907.23	33.69	29.68	188.27	357.11	2.93
TC-SB	768	1601.98	37.42	1907.18	33.70	29.68	188.27	356.46	2.93
MLM	768	1660.41	37.91	1916.34	33.53	29.82	188.27	356.70	2.93
Hybrid (MLM+TC)	768	1514.89	40.13	1831.48	35.09	28.50	188.27	356.35	2.93
Qwen3-Emb-8B	4096	4155.42	162.84	4683.61	13.72	72.88	1004.11	1218.07	15.63

Enc. = offline embedding time; Idx. = HNSW indexing time; Total = total pipeline time; Tput = encoding throughput; Emb. = embedding-array size; Qdrant = ANN index size; /1K = storage per 1,000 embeddings.

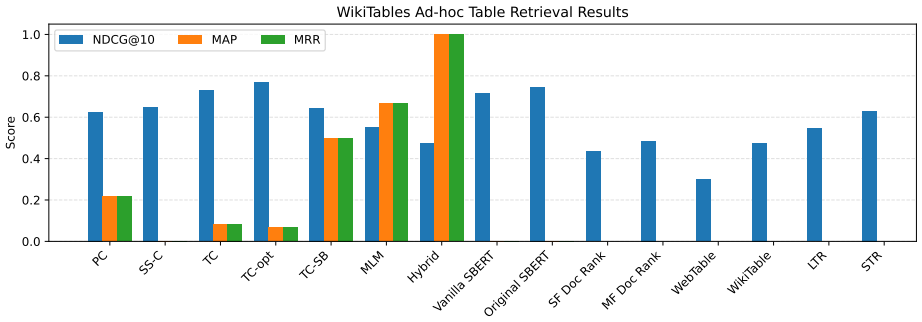

Figure 4: Ad-hoc Table Retrieval Performance

access to fine-grained table content. Structure-aware fine-tuning closes this gap by integrating relational structure with cell-level semantics, yielding higher F1 scores across TabFact.

4.3.3 Question Answering Retrieval. Table 7 reports retrieval and generation performance on FeTaQA and OTTQA, two benchmarks that stress value-grounded table relevance. In both cases, performance differences are primarily driven by serialization choices and fine-tuning objectives rather than query format alone.

Across both datasets, the Hybrid (MLM+TC) model is the most effective retrieval and downstream generation performance. This indicates that combining content-aware pretraining with supervised ranking produces representations that align well with evidence-bearing table values. The effect is most pronounced on FeTaQA, where retrieval requires identifying multi-row explanatory evidence and schema-level cues are insufficient.

Supervised contrastive strategies, including PC and triplet-based variants, achieve strong Recall@k but consistently lag behind Hybrid in BLEU. This gap suggests that while these methods often retrieve relevant tables, the retrieved evidence is less well aligned for answer generation, highlighting the importance

Table 5: Ad-hoc table retrieval results on WikiTables [51].

Model	AvgCos	MAP	MRR	NDCG@10	NDCG@20	NDCG@60
PC	0.4561	0.2167	0.2167	0.6269	0.6856	0.8961
SS-C (SimCSE)	0.1208	0.0000	0.0000	0.6493	0.7312	0.9025
TC	0.2737	0.0833	0.0833	*0.7286*	*0.7726*	0.9394
TC-opt	0.2781	0.0667	0.0667	**0.7681**	**0.8184**	**0.9520**
TC-SB	*0.5321*	*0.5000*	*0.5000*	0.6448	0.7091	0.8968
MLM	0.6746	0.6667	0.6667	0.5526	0.6233	0.8563
Hybrid (MLM+TC)	**0.9276**	**1.0000**	**1.0000**	0.4761	0.5582	0.8396
Vanilla SBERT	0.1321	0.0000	0.0000	0.7149	0.7674	0.9296
Original SBERT	0.1450	0.0000	0.0000	0.7454	0.7884	*0.9393*
SF Doc Rank [4]	–	–	–	0.4344	0.4586	–
MF Doc Rank [31]	–	–	–	*0.4860*	*0.5170*	–
WebTable [4]	–	–	–	0.2992	0.3311	–
WikiTable [2]	–	–	–	0.4766	0.5062	–
LTR [51]	–	–	–	0.5456	0.5738	–
STR [51]	–	–	–	**0.6293**	**0.6590**	–

Bold indicates best, underlined second-best, and *italic* third-best performance.

Table 6: TabFact Evidence-table Retrieval Performance

Model	Precision	Recall	F1 Score
PC	0.5525	*0.7831*	0.6479
SS-C (SimCSE)	*0.5714*	0.0074	0.0145
TC	0.5426	0.7849	*0.6416*
TC-opt	0.5496	0.7537	0.6357
TC-SB	0.5487	0.6838	0.6088
MLM	0.5671	0.5827	0.5748
Hybrid (MLM+TC)	0.5465	**0.9945**	**0.7053**
Vanilla SBERT	0.5800	0.2132	0.3118
Original SBERT	**0.5949**	0.0864	0.1509
Heuristic baseline	0.5633	0.1636	0.2536
Qwen3-Emb-8B	0.1324	0.6622	0.2207
TAPEX [25]	*0.639*	**0.697**	**0.666**
TAPAS [13]	**0.683**	*0.554*	0.612
TABBIE [15]	0.604	0.569	0.586
ReFinED [1]	0.672	0.526	*0.589*
FinQANet [9]	0.631	0.539	0.581
STR [51]	–	–	**0.6900**
LTR [51]	–	–	0.6780
Neural baseline [51]	–	–	*0.6360*
BM25 [17]	–	–	0.4380

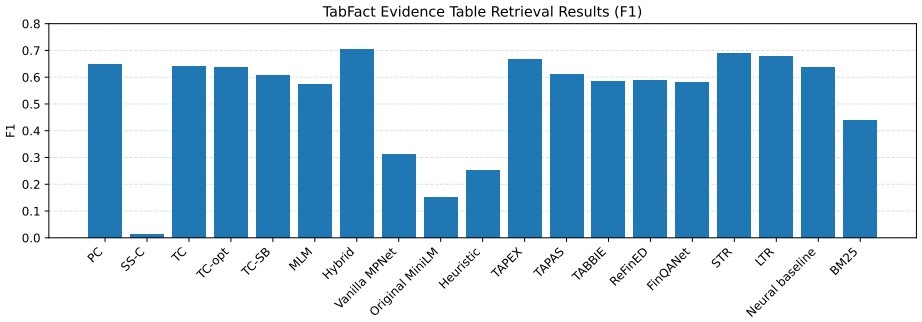

Figure 5: TabFact F1 Scores

of content-sensitive representation learning beyond coarse relevance. Self-supervised and generic sentence encoders perform substantially worse, confirming that value-level grounding is difficult to learn without explicit supervision and access to table content.

A clear contrast emerges between the two QA benchmarks. OTTQA is largely driven by entity-centric factoid retrieval, where semantic matching already provides strong recall, while FeTaQA places greater emphasis on explanatory evidence grounded in table content across multiple rows. TABERTA-Hybrid remains effective across both settings, suggesting that table-oriented fine-tuning and content-aware serialization complement general semantic matching when retrieval requires value-level and multi-row evidence.

4.3.4 Schema and Table Grounding. Tables 8,9, and Figure 6 report grounding-oriented retrieval results on Spider [48] and BIRD [49].

Across both datasets, Hybrid (MLM + TC) achieves the strongest performance, reaching CR@10 of 0.8976 on Spider and 0.9900 on BIRD. This gap relative to single-objective training is consistent with the need to combine content-sensitive representation learning (via MLM over FullView) with supervised ranking signals that stabilize query-schema alignment. Triplet-based objectives also perform strongly: TC and TC-opt provide competitive grounding accuracy, and TC-opt is strong for WikiTables NDCG, aligning with its design that explicitly incorporates SchemaView alongside FullView during optimization. This suggests that grounding is sensitive to serialization: schema-focused cues help disambiguate which tables participate in the SQL query, while value exposure alone can introduce noise when schemas are complex.

Spider is more challenging and exposes larger separations between strategies: lexical baselines (BM25/TF–IDF) plateau around CR@10 $\approx$ 0.54, and generic encoders remain far below structure-aware fine-tuning, indicating that keyword overlap is insufficient for heterogeneous schemas. BIRD is generally easier for dense retrieval, and several strong dense baselines approach high CR@10; nevertheless, Hybrid (MLM + TC) remains best, indicating that structure-aware fine-tuning still improves robustness even when dense representations already generalize well.

Decoder-Only Baseline (Qwen3-Embedding-8B). Provides a strong modern decoder-only baseline but yields mixed results across tasks. It excels on OTTQA (R@10 = 0.9684, MRR@10 = 0.8823), confirming that large decoder-only embeddings capture broad semantic similarity for open-domain entity-driven retrieval. However, it is less consistent on FeTaQA (R@10 = 0.4905, NDCG@10 = 0.3688), Spider (CR@10 = 0.6167, below the strongest TABERTA variants), and TabFact (F1 = 0.2207, high recall but low precision). These results support a balanced conclusion: TABERTA's table-oriented serialization and retrieval fine-tuning remain important for structure-sensitive and evidence-grounded settings where schema cues, relational context, and cell-level value matching drive relevance. The efficiency comparison (Table 4) further shows that Qwen3's 4096-dimensional embeddings require 2.7× longer encoding and produce indexes over 3× larger than TABERTA-Hybrid, making TABERTA more practical for large-scale offline indexing.

Who Wins When: Across all five benchmarks, **Hybrid (MLM+TC)** delivers the strongest overall performance, combining MLM pre-training with triplet-contrastive fine-tuning. Table 10 provides a per-scenario guide for cases where a lighter or more specialized strategy is preferable.

4.3.5 Evaluation Conclusion. TABERTA supports effective table retrieval; we illustrate its end-to-end behaviour on the motivating Clooney example (Figure 1). The query "*What are George Clooney's best movies?*" distributes the relevant evidence across three tables from Figure 1 that are individually incomplete: a *movies* table (`title`, `rating`, `year`), a *cast* table (`movie_id`, `actor_id`), and an *actors* table (`actor_id`, `actor_name`). Through schema- and content-aware serialization views, TABERTA embeds each table in a manner that captures both its semantic focus and its structural role within the dataset. Concretely, the Hybrid model retrieves the *movies* table at rank 1 via content matching (`title` and `rating` columns directly encode "best movies" semantics), the *actors* table at rank 3 via entity grounding (`actor_name` creates a semantic link to the query entity "George Clooney"), and the *cast* table at rank 5 via structural schema awareness, despite containing only foreign-key identifiers (`movie_id`, `actor_id`) with no natural-language content, TABERTA's relational serialization recognizes it as the bridge table connecting movies to actors (Table B in Figure 1). All three complementary tables are

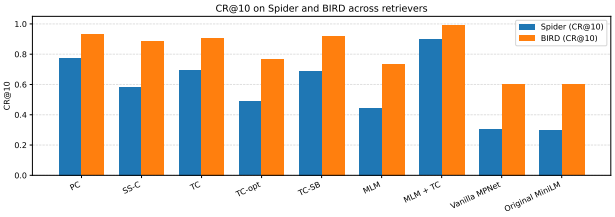

Figure 6: CR@10 across retrievers on Spider and BIRD.

Table 7: Retrieval and generation performance on FeTaQA and OTTQA validation sets, baselines ranked by R@10.

Model	FeTaQA / BART				FeTaQA / T5				OTTQA / BART				OTTQA / T5			
	R@1	R@5	R@10	BLEU	R@1	R@5	R@10	BLEU	R@1	R@5	R@10	BLEU	R@1	R@5	R@10	BLEU
PC	0.550	0.750	0.850	6.0	0.500	0.700	0.800	20.0	0.600	0.800	0.900	7.5	0.600	0.800	0.900	3.75
SS-C (SimCSE)	0.350	0.550	0.650	3.0	0.250	0.450	0.550	14.0	0.380	0.580	0.680	3.0	0.380	0.580	0.680	1.50
TC	*0.480*	*0.680*	*0.780*	*5.0*	*0.400*	*0.600*	*0.720*	*16.5*	*0.520*	*0.720*	*0.820*	*6.0*	*0.520*	*0.720*	*0.820*	*3.00*
TC-opt	0.460	0.660	0.760	4.5	0.380	0.580	0.680	16.0	0.500	0.700	0.800	5.5	0.500	0.700	0.800	2.75
TC-SB	0.500	0.700	0.800	5.5	0.450	0.650	0.750	18.0	0.550	0.750	0.850	6.5	0.550	0.750	0.850	3.25
MLM	0.400	0.600	0.700	4.0	0.300	0.500	0.620	15.5	0.450	0.650	0.750	5.0	0.450	0.650	0.750	2.50
Hybrid (MLM+TC)	**0.980**	**0.990**	**1.000**	**7.0**	**0.950**	**0.980**	**0.990**	**23.0**	**0.970**	**0.990**	**1.000**	**9.0**	**0.970**	**0.990**	**1.000**	**4.50**
Vanilla SBERT	0.380	0.580	0.680	3.5	0.280	0.480	0.600	15.0	0.420	0.620	0.720	4.0	0.420	0.620	0.720	2.00
Original SBERT	0.370	0.570	0.670	3.4	0.270	0.470	0.590	14.8	0.400	0.600	0.700	3.8	0.400	0.600	0.700	1.90
Qwen3-Emb-8B	0.2597	0.4266	0.4905	–	0.2597	0.4266	0.4905	–	0.8293	0.9526	0.9684	–	0.8293	0.9526	0.9684	–

Table 8: Schema/table grounding retrieval results (top-k containment recall and ranking metrics).

	Model	CR@1	CR@5	CR@10	CappedR@10	MRR@10	NDCG@10
Spider	PC	0.7100	0.7350	0.7750	0.7750	0.7381	0.7434
	SS-C (SimCSE)	0.5092	0.5478	0.5850	0.5850	0.5640	0.5690
	TC	*0.6188*	*0.6657*	*0.6947*	*0.6947*	*0.6693*	*0.6767*
	TC-opt	0.4222	0.4304	0.4884	0.4884	0.4773	0.4803
	TC-SB	0.6043	0.6464	0.6899	0.6899	0.6547	0.6675
	MLM	0.4159	0.4436	0.4449	0.4449	0.4735	0.4895
	Hybrid (MLM+TC)	**0.8188**	**0.8802**	**0.8976**	**0.8976**	**0.8901**	**0.8976**
	Vanilla SBERT	0.1884	0.2705	0.3043	0.3043	0.3101	0.3090
	Original SBERT	0.1990	0.2640	0.3010	0.3010	0.2999	0.3027
	Qwen3-Emb-8B	0.2757	0.5562	0.6167	0.6167	0.5450	0.5526
BIRD	PC	0.9060	0.9178	0.9302	0.9302	0.8222	0.9423
	SS-C (SimCSE)	0.7752	0.8348	0.8840	0.8840	0.7760	0.8960
	TC	*0.8848*	0.8927	0.9081	0.9081	0.8005	0.9201
	TC-opt	0.6882	0.7174	0.7674	0.7674	0.6594	0.7794
	TC-SB	0.8703	*0.9034*	*0.9189*	*0.9189*	*0.8109*	*0.9309*
	MLM	0.7027	0.7095	0.7333	0.7333	0.6259	0.7459
	Hybrid (MLM+TC)	**0.9841**	**0.9899**	**0.9900**	**0.9900**	**0.9602**	**0.9704**
	Vanilla SBERT	0.4544	0.5575	0.6033	0.6033	0.4953	0.5153
	Original SBERT	0.4650	0.5510	0.6000	0.6000	0.4920	0.5120

CappedR@10≡CR@10 on both datasets as each query has at most one gold schema. Qwen3-Emb-8B reported on Spider only (no BIRD run completed)

recovered, demonstrating that structure-aware embeddings surface both direct-match tables and indirect bridge tables without requiring explicit join modeling. Importantly, this behavior does not arise from explicitly modeling joinability or enforcing table-to-table links at retrieval time. Rather, by training the encoder to understand table content while respecting structural cues, TABERTA learns embeddings that implicitly reflect how tables can complement each other semantically.

5 Related Work

[Embedding-Based Table Representation Learning] Prior work on table representation learning adapts language models to tabular data and achieves strong performance on downstream tasks such as table question answering and entity-centric reasoning. Transformer-based models (TURL [10], TaBERT [47], TAPAS [13]) and subsequent structure-aware approaches (e.g., TABBIE [15], RPT [41]) incorporate schema and relational signals through linearization, self-supervision, or graph-based modeling. More recent work leverages large relational corpora such as WikiDBs [45] to pretrain structure-aware representations at scale, including contrastive and graph-based objectives [23, 24]. Despite these advances, existing methods are optimized for task-specific reasoning or single-table settings, and do not target reusable table embeddings for retrieval-oriented dataset discovery across heterogeneous schemas.

[Joinability and Unionability-Oriented Dataset Discovery] A related line of work in dataset discovery focuses on identifying joinable or unionable tables based on schema compatibility, column overlap, or value similarity. Unionability aims to detect tables with compatible schemas that can be vertically merged, while joinability targets tables that can be linked through shared attributes or keys [22, 38].

To improve scalability, several systems employ approximate set similarity and indexing techniques to detect candidate joins or unions over large data lakes [39]. While effective as filtering mechanisms, these methods largely operate independently of user queries and do not model semantic relevance beyond column-level overlap. More recent approaches incorporate learned representations to improve robustness under naming variation and noisy data [52], but they remain primarily optimized for structural compatibility rather than relevance-driven retrieval.

Table 9: TARGET CR@10 Performance

Baseline	Spider CR@10	BIRD CR@10
BM25 (Sparse Lexical)	0.544	0.700
TF-IDF (Sparse Lexical)	0.541	0.586
Dense Table Embedding	0.657	0.961
Dense Row-level Embedding	0.665	0.951
Dense Metadata Embedding	0.621	0.940
text-embedding-3-small	0.618	0.858
stella_en_400M_v5	**0.657**	**0.961**
multilingual-e5-large-instruct	0.620	0.894

Table 10: Recommended TABERTA Configuration

Retrieval Scenario	Best Strategy	Rationale
General-purpose	Hybrid (MLM+TC)	Best overall across benchmark groups
Ranking depth (NDCG)	TC-opt	Strong ranking-depth behavior on WikiTables
Schema/table grounding	Hybrid (MLM+TC)	Best CR@10 on Spider and BIRD
Evidence lookup	Hybrid (MLM+TC)	Highest TabFact F1 and strong QA retrieval
Low-compute budget	TC or TC-opt	Avoids staged MLM+TC training

Large-scale benchmarks such as LakeBench [11] highlight both the progress and the limitations of joinability- and unionability-focused methods, particularly their difficulty in capturing nuanced semantic relationships across heterogeneous datasets.

[Dataset Discovery via Lexical and Semantic Matching] A parallel line of work studies dataset discovery and table search as a retrieval problem, where relevant datasets are identified by matching user queries against dataset metadata and structural descriptors. Early systems rely primarily on lexical similarity between queries and metadata fields such as table titles, column names, and textual descriptions, using sparse retrieval models including TF–IDF and BM25 [36, 37]. These approaches fit keyword-driven search but are sensitive to vocabulary mismatch and metadata quality.

To address these limitations, later work combines lexical retrieval with explicit semantic signals in multi-stage pipelines. Such systems typically use keyword-based retrieval for candidate generation, followed by semantic reranking based on learned features, embeddings, or schema-aware similarity measures [17, 51]. By integrating lexical and semantic matching, these pipelines improve recall and ranking robustness compared to purely lexical methods, particularly when schema information is informative.

However, pipeline-based combinations of lexical and semantic signals often depend on task-specific feature engineering, curated metadata, or external resources whose availability and consistency vary across data lakes. As a result, retrieval effectiveness can degrade in heterogeneous or weakly annotated settings, motivating approaches that learn reusable representations capable of supporting dataset discovery with minimal manual design [17, 51].

[Joinability and Unionability-Oriented Dataset Discovery] Joinability and unionability identify structurally compatible tables for joins or unions using schema overlap, shared attributes, value similarity, and attribute-linkage signals [22, 38, 43]. Early scalable approaches rely on syntactic, set-based, or approximate similarity measures over columns and values [39, 46], while recent methods use learned representations to improve robustness under naming variation and noise [52].

[Decoder-Only Embedding Models] Recent general-purpose embedding models based on decoder-only architectures have demonstrated strong performance on semantic retrieval benchmarks. Models such as Qwen3-Embedding [42] generate high-dimensional (4096-dim) embeddings from large pre-trained language models and achieve competitive results on standard text and document retrieval tasks. However, these models are trained as frozen general-purpose retrievers without structural adaptation to relational tables and incur substantially higher storage and indexing cost due to their embedding dimensionality. TABERTA addresses the complementary objective of efficient bidirectional table retrieval, combining structure-aware serialization with retrieval-oriented fine-tuning to produce compact 768-dimensional embeddings that capture both schema signals and cell-level evidence.

[Gap and Positioning] Prior work leaves a gap between dataset discovery and table representation learning: discovery methods often rely on lexical or semantic matching over metadata, while structure-aware table models are usually optimized for task-specific reasoning rather than reusable retrieval. TABERTA addresses this gap by learning compact table embeddings that encode schema, content, and relational cues through multiple serialization views, enabling semantic retrieval across heterogeneous table collections.

6 Conclusion

We presented a structure-aware fine-tuning framework for relational table retrieval. It adapts a bi-encoder through complementary serialization views and retrieval-oriented objectives, allowing table embeddings to capture schema structure, value-level evidence, and relational context rather than treating tables as flat text. Across heterogeneous benchmarks, our results show that serialization is a critical design choice: schema-focused views are effective for ad-hoc schema search, while content-aware and hybrid views are needed when relevance depends on value grounding or evidence distributed across tables. Hybrid (MLM+TC) provides the strongest general-purpose configuration across evidence selection, fact verification, question-answering retrieval, and schema/table grounding, while lighter triplet variants remain useful when training cost is constrained.

TABERTA also shows that retrieval quality must be considered together with deployment cost. Its offline-to-online design computes table embeddings once, indexes them with ANN search, and reuses them at query time. This makes the framework practical for large table repositories: compared with the evaluated high-dimensional decoder-only baseline, TABERTA reduces offline pipeline time by 60.9% and ANN index size by 3.4×. Fine-tuned TABERTA models and resources are publicly available on GitHub[1] and Hugging Face[2].

References

[1] Tom Ayoola, Shubhi Tyagi, Joseph Fisher, Christos Christodoulopoulos, and Andrea Pierleoni. 2022. ReFinED: An Efficient Zero-shot-capable Approach to End-to-End Entity Linking. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*. 209–220.

[2] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. 2013. Methods for exploring and mining tables on Wikipedia. In *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics*. 18–26.

[3] Allaa Boutaleb, Bernd Amann, Hubert Naacke, and Rafael Angarita. 2025. A Critical Re-evaluation of Table Union Search Benchmarks. In *Proceedings of the 4th Table Representation Learning Workshop*. 71–85.

[4] Michael J Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. WebTables: exploring the power of tables on the web. In *Proceedings of the VLDB Endowment*. 538–549.

[5] Riccardo Cappuzzo, Paolo Papotti, and Saravanan Thirumuruganathan. 2021. EmbDI: Generating Embeddings for Relational Data Integration (Discussion Paper). In *Proceedings of the 29th Italian Symposium on Advanced Database Systems*. 331–338.

[6] Cheng Chen, Zhu Xie, and Xiaoyu Yuan. 2020. TabFact: A Large-scale Dataset for Table-based Fact Verification. In *International Conference on Learning Representations (ICLR)*.

[7] Daya Guo Chen and et al. 2020. Open Table and Text Question Answering. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[8] Tianqi Chen, Bing Xu, Chiyuan Zhang, Zhiyuan Zhang, and Mu Li. 2016. Training Deep Nets with Sublinear Memory Cost. *CoRR* abs/1604.06174 (2016).

[9] Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Kenneth Huang, Bryan R. Routledge, and William Yang Wang. 2021. FinQA: A Dataset of Numerical Reasoning over Financial Data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 3697–3711.

[10] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2020. TURL: Table Understanding through Representation Learning. *Proc. VLDB Endow.* 14, 3 (2020), 307–319.

[11] Yuhao Deng, Chengliang Chai, Lei Cao, et al. 2024. LakeBench: A Benchmark for Discovering Joinable and Unionable Tables in Data Lakes. *Proceedings of the VLDB Endowment* 17, 8 (2024), 1925–1938.

[12] Huang Fang. 2015. Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem. In *2015 IEEE International Conference on Cyber Technology in Automation*. 820–824.

[13] Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. TaPas: Weakly Supervised Table Parsing via Pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 4320–4333.

[14] Xuming Hu, Xiao Qin, Chuan Lei, Asterios Katsifodimos, Zhengyuan Shen, Balasubramaniam Srinivasan, and Huzefa Rangwala. 2025. DiscoverGPT:

[1] https://github.com/enaskhwaileh/TABERTA.git
[2] https://huggingface.co/EnasKhwaileh/TABERTA

Multi-task Fine-tuning Large Language Model for Related Table Discovery. In *NAACL 2025*. 358–373.

[15] Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. TABBIE: Pretrained Representations of Tabular Data. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 3446–3456.

[16] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Transactions on Information Systems* 20, 4 (2002), 422–446.

[17] Xingyu Ji, Parker Glenn, Aditya G. Parameswaran, and Madelon Hulsebos. 2025. TARGET: Benchmarking Table Retrieval for Generative Tasks. In *Proceedings of the VLDB Endowment (VLDB)*.

[18] Aamod Khatiwada, Harsha Kokel, Ibrahim Abdelaziz, Subhajit Chaudhury, Julian Dolby, Oktie Hassanzadeh, Zhenhan Huang, Tejaswini Pedapati, Horst Samulowitz, and Kavitha Srinivas. 2025. TabSketchFM: Sketch-Based Tabular Representation Learning for Data Discovery Over Data Lakes. In *ICDE 2025, Hong Kong, May 19-23, 2025*. IEEE, 1523–1536.

[19] Enas Khwaileh and Yannis Velegrakis. 2025. Dataset Discovery using Semantic Matching. In *Proceedings 28th International Conference on Extending Database Technology*. 649–660.

[20] Eddine Laouir, Ala, Imine, and Abdessamad. 2024. Private Approximate Query over Horizontal Data Federation. (2024).

[21] 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 *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1142–1151.

[22] Xiaodong Li et al. 2020. D3L: Deep Learning Enhanced Discovery of Unionable Tables in Data Lakes. *Proceedings of the VLDB Endowment* 13, 5 (2020), 635–647.

[23] Xinyang Li, Jing Liu, Jianfeng Gao, et al. 2024. Unifying Structured Data as Graphs for Pretraining. *(TACL)* (2024).

[24] Xinyang Li, Jing Liu, Weijia Xu, Fan Yang, and Yue Zhang. 2023. Structure-Aware Pretraining for Dense Retrieval. In *(ACL)*.

[25] Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2022. TAPEX: Table Pre-training via Learning a Neural SQL Executor. In *The Tenth International Conference on Learning Representations*.

[26] Yury A. Malkov and Dmitry A. Yashunin. 2020. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[27] Christian Mathis. 2017. Data lakes. *Datenbank-Spektrum* 17, 3 (2017), 289–293.

[28] Paulius Micikevicius, Sharan Narang, Jonah Alben, Greg Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed Precision Training. In *International Conference on Learning Representations (ICLR)*.

[29] Chang Nan, Shuohang Wang, and et al. 2022. FeTaQA: Freeform Table Question Answering. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[30] Fatemeh Nargesian, Erkang Zhu, Renée J Miller, Ken Q Pu, and Patricia C Arocena. 2019. Data lake management: challenges and opportunities. *VLDB* 12, 12 (2019), 1986–1989.

[31] Rakesh Pimplikar and Sunita Sarawagi. 2012. Answering table queries on the web using column keywords. In *Proceedings of the VLDB Endowment*. 908–919.

[32] Matt Post. 2018. A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation (WMT)*. 186–191.

[33] Qdrant. 2026. Qdrant: Vector Similarity Search Engine and Vector Database. https://github.com/qdrant/qdrant. Accessed: 2026-04-28.

[34] 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*. 3980–3990.

[35] 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 (EMNLP-IJCNLP)*. Association for Computational Linguistics, 3982–3992.

[36] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval* 3, 4 (2009), 333–389.

[37] Gerard Salton and Christopher Buckley. 1988. Term-weighting Approaches in Automatic Text Retrieval. *Information Processing & Management* 24, 5 (1988), 513–523.

[38] Daniel Santos et al. 2017. Discovering Unionable Tables. *Proceedings of the VLDB Endowment* 10, 12 (2017), 1818–1829.

[39] Jianfeng Shang, Jiahui Jin, Yuliang Li, and Ji-Rong Wen. 2019. LSH Ensemble: Scaling Minhash for Finding Joinable Columns in Data Lakes. *Proceedings of the VLDB Endowment* 12, 9 (2019), 1116–1128.

[40] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. MPNet: Masked and Permuted Pre-training for Language Understanding. In *s (NeurIPS)*.

[41] Nan Tang, Ju Fan, Fangyi Li, Jianhong Tu, Xiaoyong Du, Guoliang Li, Samuel Madden, and Mourad Ouzzani. 2021. RPT: Relational Pre-trained Transformer Is Almost All You Need towards Democratizing Data Preparation. *Proc. VLDB Endow.* 14, 8 (2021), 1254–1261.

[42] Qwen Team. 2025. Qwen3-Embedding: Advancing Text Embedding with Decoder-Only Architecture. https://huggingface.co/Qwen/Qwen3-Embedding.

[43] Leonard Traeger, Andreas Behrend, and George Karabatis. 2025. SEALM: Semantically Enriched Attributes with Language Models for Linkage Recommendation:. In *27th ICEIS*. Porto, Portugal, 39–50.

[44] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. In *Advances in Neural Information Processing Systems*.

[45] Liane Vogel, Jan-Micha Bodensohn, and Carsten Binnig. 2024. WikiDBs: A Large-Scale Corpus Of Relational Databases From Wikidata. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024*.

[46] Jun Wang et al. 2018. Josie: Efficient Joinability Search Across Large Data Lakes. *Proceedings of the SIGMOD Conference* (2018), 343–355.

[47] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 8413–8426.

[48] Tao Yu and et al. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Text-to-SQL Semantic Parsing. In *EMNLP*.

[49] Ji Zhang and et al. 2023. BIRD: Benchmark for Information Retrieval from Databases. In *VLDB*.

[50] Li Zhang, Shuo Zhang, and Krisztian Balog. 2019. Table2Vec: Neural Word and Entity Embeddings for Table Population and Retrieval. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1029–1032.

[51] Shuo Zhang and Krisztian Balog. 2018. Ad hoc table retrieval using semantic similarity. In *Proceedings of the 2018 world wide web conference*. 1553–1562.

[52] Xiaoying Zhu et al. 2020. DeepJoin: Learning to Discover Joinable Tables in Data Lakes. *Proceedings of the VLDB Endowment* 13, 10 (2020), 1568–1581.