Tailoring Table Retrieval from a Field-aware Hybrid Matching Perspective

Anonymous ACL submission

Abstract

Table retrieval, essential for accessing information through tabular data, is less explored compared to text retrieval. The row/column structure and distinct fields of tables (including titles, headers, and cells) present unique challenges. For example, different table fields have varying matching preferences: cells may favor finer-grained (word/phrase level) matching over broader (sentence/passage level) matching due to their fragmented and detailed nature, unlike titles. This necessitates a table-specific retriever to accommodate the various matching needs of each table field. Therefore, we introduce a **T**able-tailored **HY**brid **M**atching r**E**triever (**THYME**), which approaches table retrieval from a field-aware hybrid matching perspective. Empirical results on two table retrieval benchmarks, NQ-TABLES and OTT-QA, show that THYME significantly outperforms state-of-the-art baselines. Comprehensive analyses have confirmed the differing matching preferences across table fields and validated the efficacy of THYME.

1 Introduction

Table retrieval is an important way for seeking information stored in tables, organized in rows and columns (Cafarella et al., 2008; Jauhar et al., 2016; Zhang and Balog, 2020). Its significance is evident in real-world applications: for instance, in the Natural Questions dataset constructed from Google queries targeting Wikipedia pages, table-based information needs account for 25.6% of all questions (Kwiatkowski et al., 2019). Retrieved tables serve as the input of table-related tasks such as question answering (Cafarella et al., 2008; Jauhar et al., 2016) and fact verification (Chen et al., 2020b). Despite extensive studies on unstructured text retrieval, structured table retrieval remains under-explored. We aim to enhance table retrieval performance to better serve table-related information needs.

Table retrieval presents unique challenges compared to text retrieval: (1) Text data are usually unstructured, while tables have a structured format with cells, rows, columns, and headers, suggesting a table-specific encoding approach. (2) Unlike documents whose sentence order matters, tables' data entry order does not affect their information. (3) Each row in a table is equally important, making it challenging to compress table information into dense representations. (4) Table cells contain detailed information, often in words or phrases, making local finer-grained matching more critical than in document retrieval. Figure 1 shows an example to illustrate the importance of local lexical matching in table retrieval.

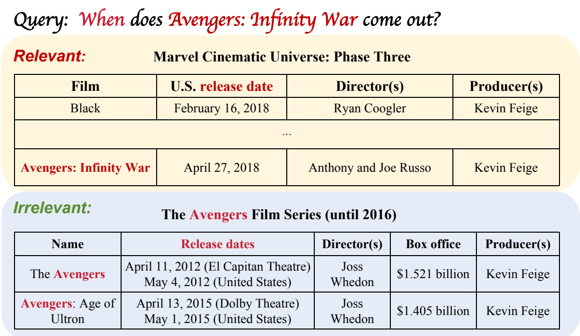

Figure 1: A case of table retrieval. It shows fine-grained matching in cells is important.

Current table retrieval methods have investigated various strategies, including: (1) condensing table information by selecting cells (Herzig et al., 2020) and row/column aggregation (Trabelsi et al., 2022) and (2) specialized pre-training objectives for table retrieval (Herzig et al., 2021; Chen et al., 2023). However, these methods often compress tables into dense representations, which may not capture the semantics of parallel data rows and fine-grained exact matching effectively.

We approach table retrieval from a field-aware hybrid matching perspective, integrating both sparse and dense representations to adapt effectively to various table fields. Sparse representations (Formal et al., 2021) preserve detailed token-level information, complementing the global semantics carried in dense representations, which are particularly well-suited for table cells. In contrast, dense representations excel in handling unstructured text fields, such as titles, aligning with proven effectiveness in passage retrieval (Guo et al., 2025). A key challenging issue is how to adaptively learn the optimal representation and matching pattern for each table field.

To this end, we introduce **THYME**, a **T**able-tailored **HY**brid **M**atching r**E**triever for field-aware hybrid matching. Using a shared encoder, we construct dense and sparse representations for queries and tables. The [CLS] embedding implicitly captures field importance/preference on coarse-grain semantics through extensive Transformer interactions. In contrast, sparse representations hold greater potential for field-specific considerations due to their explicit segmentation of content tokens by field. Therefore, we focus on learning field-aware sparse representations, which can reflect field importance on fine-grained semantics. Based on a shared encoder, sparse and dense representations can be learned coordinately and finalize the field suitability for coarse and fine grains of semantics during relevance matching. Specifically, for sparse representations of table bodies (headers and cells), we employ mean pooling to retain similar types of information within columns and max pooling to extract the most important semantics across columns. Then, we learn the field (title, header, cell) importance of each token/dimension in the sparse representation, and aggregate them dynamically for matching. The final relevance score is computed as the sum of dense and sparse matching scores. During training, we use a score dropout strategy to enable adaptive learning across both lexical (sparse) and semantic (dense) matching pathways.

We evaluate THYME on table retrieval benchmarks, NQ-TABLES (Herzig et al., 2021) and OTT-QA (Chen et al., 2020a), showing that it outperforms state-of-the-art baselines, including sparse, dense, and hybrid retrievers. Analyses confirm that table titles prefer dense matching, while headers and cells prefer sparse matching, and THYME effectively captures these preferences. Within the RAG framework, THYME enhances the results of various LLMs by providing more relevant tables. Our studies indicate a promising way of elevating table retrieval, which can shed light on future research on this topic.

2 Related Work

2.1 Text Retrieval

Text retrievers can be classified into three types based on representations used: sparse, dense, and hybrid retrieval which incorporates them.

Sparse retrievers refer to models that use sparse representations such as TF-IDF (Sparck Jones, 1972) and BM25 (Robertson and Walker, 1994). Building on pre-trained language models (PLMs), SparTerm (Bai et al., 2020) and SPLADE (Formal et al., 2021) aim to generate term distributions over vocabulary. Dense retrievers typically encode inputs to dense vectors using PLMs (Devlin et al., 2019; Liu et al., 2019). Dense and Sparse retrieval have complementary advantages. Dense retrieval generally outperforms sparse retrieval, while the latter can be more effective when limited training data is available. Hybrid retrieval can combine the advantages of them (Craswell et al., 2020; Bajaj et al., 2018). A straightforward combining approach is to train two different types of retrievers independently and then combine their outputs linearly to give a final relevance score(Chen et al., 2021; Kuzi et al., 2020; Lin and Lin, 2021; Luan et al., 2020; Guo et al., 2025; Shen et al., 2023). There are some other ways to combine different retrievers such as: CLEAR (Gao et al., 2020) utilizing boosting, UnifieR (Shen et al., 2023) leveraging knowledge distillation.

2.2 Table Search

Table search has emerged as a fundamental research challenge in structured data search(Cafarella et al., 2008; Zhang and Balog, 2018; Bhagavatula et al., 2013). To maintain both efficiency and effectiveness, the table search is typically decomposed into two phases: retrieval and reranking.

For table retrieval, PLMs exhibit limited performance due to their primary training on textual corpora. To improve PLMs' comprehension of tabular structures, TAPAS (Herzig et al., 2020) and DTR (Herzig et al., 2021) utilize distinct types of embeddings like row and column embeddings, to represent the structure of tables based on BERT (Devlin et al., 2019). Alternatively, fine-tuning PLMs on the table corpus can improve their comprehension of tables like UTP (Chen et al., 2023). Given the complexity of table structure and content, a single dense representation often fails to capture fine-grained details. SSDR (Jin et al., 2023) aims to represent both the query and table through multiple vector representations and graph-based methods (Wang et al., 2021) have also been adapted for table retrieval.

Table reranking fundamentally differs from table retrieval through its joint encoding of query-table pairs, enabling more sophisticated interaction than the independent processing of retrieval. Existing approaches such as TaBERT (Yin et al., 2020), StruBERT (Trabelsi et al., 2022) and others (Shraga et al., 2020) improve results by emphasizing structural information during encoding.

Both retrieval and reranking tasks critically depend on table structure representation, our work takes a fundamentally different approach from existing complex architectures. Rather than designing elaborate structural encoding methods, we investigate domain-specific matching preferences to optimize table retrieval performance.

3 Task Description

Let $D = \{(q_i, T_i^+)\}_{i=1}^N$ be a labeled dataset, where q_i denotes an individual query and T_i^+ is a set of tables $\{t_i^+\}$ that are considered relevant to q_i with variable size per query. Table retrieval aims to train a retriever to learn query-table relevance matching considering table structure and fields. After training, this retriever is expected to understand how the fields $F = \{title, headers, cells\}$ align with the information needs expressed in the query. Ultimately, for any query q, the retriever can retrieve relevant tables from a given collection.

4 Table-Tailored Hybrid Matching Retriever (THYME)

In this section, we introduce **THYME**, a **T**able-tailored **HY**brid **M**atching r**E**triever. Figure 2 illustrates the overall architecture. The model employs dual representations: dense representations capture semantic information from unstructured text (e.g., titles), and sparse representations preserve details for fine-grained information needs. To effectively capture the relevance matching patterns (lexical, semantic, and at various granularities) across different fields, we incorporate matching preferences of different fields, propose a field-aware lexical matching mechanism, and craft a hybrid training strategy. Note that we employ BIBERT (Lin et al., 2021) and SPLADE (Formal et al., 2021) as the backbones to calculate dense and sparse representations. Although other advanced backbones can be alternatives to achieve better performance, our focus is to study table-specific hybrid matching. Next, we detail each component of THYME.

4.1 Query Representation

Since query q is an unstructured text without special processing, we directly obtain its dense and sparse representation based on BIBERT (Lin et al., 2021) and SPLADE (Formal et al., 2021) respectively. The hidden state of [CLS] is represented as the dense representation. The sparse representation is obtained by applying max pooling over the entire sequence:

$$\begin{aligned} &\mathbf{H}_q = Enc(q),\ \mathbf{Z}_q = Trans(\mathbf{H}_q), \\ &\mathbf{q}_{cls} = \mathbf{H}_q[\text{CLS}], \\ &\mathbf{q}_{lex} = \max_{i \in |q|} \log(1 + ReLU(\mathbf{W}_q[i])), \end{aligned} \tag{1}$$

where $\mathbf{q}_{cls} \in \mathcal{R}^h$ and $\mathbf{q}_{lex} \in \mathcal{R}^{|V|}$ represent the dense and sparse query representations, respectively, h is the dimension of outputs yielded by Pre-trained Language Models (PLMs), $|V|$ is the size of the vocabulary used. $Trans(\cdot)$ is a linear used to map the output of $Enc(\cdot)$ to the distribution in the vocabulary space.

4.2 Table Serialization

To encode tables with PLMs, we explicitly annotate structural components (titles, headers, and cells) using special tokens $[TTL]$, $[HEAD]$, and $[CELL]$ to maintain their distinct semantic and structural roles. Given a table t with a $title$, n headers - $headers_n$, and $cell_{m\times n}$ of m rows and n columns, we serialize the table structure as follows:

$$\begin{aligned} t = [\,&[CLS], [TTL], title, [HEAD], header_0, \\ &\ldots header_{n-1}, [CELL], cell_{0,0}, cell_{0,1}, \ldots, \\ &cell_{m-1,n-1}, [SEP]\,]. \end{aligned}$$

4.3 Global Semantic Matching

The global semantics of a table, which encapsulates its comprehensive information by integrating all field-level data, are crucial for accurately addressing topic-related queries. With the field indicator tokens in the input sequence marking the field boundary, the self-attention mechanism enables [CLS] embedding to aggregate the information stored in each field as the global dense representation. The dense representation of a table t is

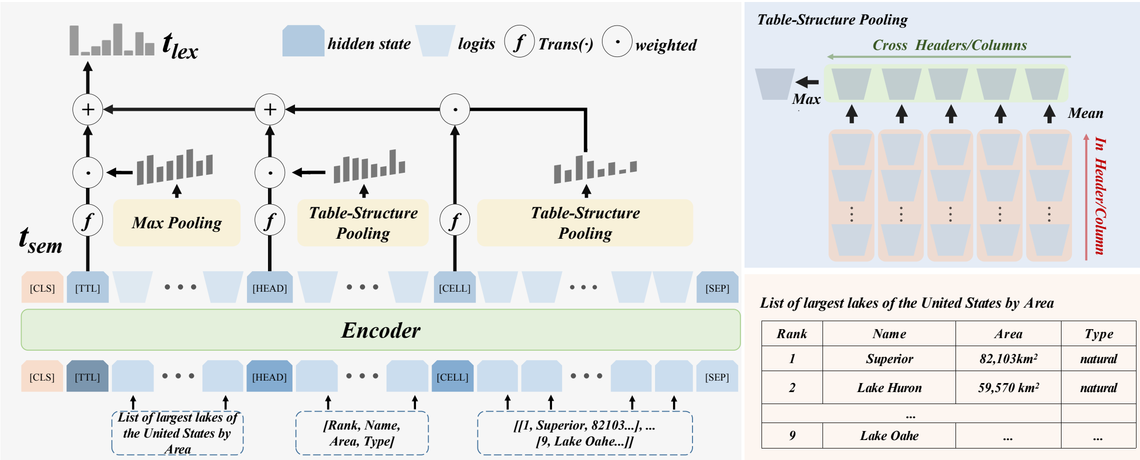

Figure 2: Illustration of THYME. Headers can be regarded as a row of cells. In this context, we adopt the identical pooling strategy that is applied to the cells. The query is treated in the same manner as the title of the table.

generated as follows:

$$\mathbf{t}_{cls} = Enc(t)[\text{CLS}]. \tag{2}$$

We have also tried alternative pooling strategies (mean and max over the entire sequence). However, it does not perform better than simply using [CLS]. The semantic matching scores between the query and the table are obtained in the following way:

$$s_{sem}(q, t) = sim(\mathbf{q}_{cls}, \mathbf{t}_{cls}). \tag{3}$$

We use the inner product as the similarity function for semantic matching.

4.4 Field-aware Lexical Matching

The body of the table, including headers and cells, mainly consists of words or phrases that lack coherent semantics. We hope to construct sparse representations for each field and obtain the final sparse representations of tables with differentiated emphasis on these fields. To this end, we propose a table-structure pooling mechanism and a mixture-of-field-experts mechanism for field-level aggregation to facilitate fine-grained lexical matching.

Table-Structure Pooling. First, to obtain sparse representations, the table t is transformed into a sequence of logits $\mathbf{Z}_t \in \mathcal{R}^{|t|*|V|}$:

$$\mathbf{H}_t = Enc(t), \mathbf{Z}_t = Trans(\mathbf{H}_t), \tag{4}$$

where $|t|$ is the length of tables and $|V|$ is the number of tokens used by PLMs. Then, we employ distinct pooling strategies for different table fields. Max pooling excels at extracting fine-grained features while mean pooling preserves every piece of information in the sequence. Based on this presumption, for the $title$, $headers$, and $cells$, our pooling strategies are as follows.

Title: Since the table title is unstructured text, similar to queries, we use max pooling as in Formal et al. (2021):

$$\mathbf{title}_{lex} = \max_{t[i] \in title} \log\left(1 + ReLU(\mathbf{Z}_t[i])\right), \tag{5}$$

where i is the index of a token within the title of table t.

Headers: Headers encode the relational schema of tabular data. We use mean pooling for the tokens within each header and max pooling across all headers to construct the sparse header representation:

$$\begin{aligned} \mathbf{header}^j_{lex} &= \operatorname*{mean}_{t[i] \in header^j} \log\left(1 + ReLU(\mathbf{Z}_t[i])\right), \\ \mathbf{headers}_{lex} &= \max_{1 \le j \le n} \mathbf{header}^j_{lex}, \end{aligned} \tag{6}$$

where $header^j$ is the j-th header among n headers in table t.

Cells: Cells in the same column share identical properties indicated by the corresponding header. The semantics carried in each cell within a column are equally important to represent the column, so we first aggregate cell-level information within each column through mean pooling. In contrast, different columns are of different importance during matching. For cross-column aggregation, we employ max pooling over column representations to emphasize discriminative features. The process can be formalized as:

$$\begin{aligned} \mathbf{col}^j_{lex} &= \operatorname*{mean}_{t[i] \in col^j} \log\left(1 + ReLU(\mathbf{Z}_t[i])\right), \\ \mathbf{cells}_{lex} &= \max_{1 \le j \le n} \mathbf{col}^j_{lex}, \end{aligned} \tag{7}$$

where $col^j = cell_{0,j}, cell_{1,j}, \cdots, cell_{m-1,j}$ is the j-th column of t which have m cells.

Mixture of Field Experts (MoFE). The sparse representation of each field computed based on Equation (5), (6), and (7), is a distribution over the vocabulary tokens. The importance of each token varies across different fields in representing the table. We adopt a Mixture of Field Experts (MoFE) mechanism to adaptively aggregate different field sparse representations. Specifically, we use the hidden states of $[TTL]$, $[HEAD]$, and $[CELL]$ to assess the importance of each token in the distribution corresponding to different fields during matching. The final sparse representation t_{lex} is calculated according to:

$$\begin{aligned} \mathbf{t}_g &= [\mathbf{H}_t[TTL], \mathbf{H}_t[HEAD], \mathbf{H}_t[CELL]], \\ \mathbf{t}_f &= [\mathbf{title}_{lex}, \mathbf{headers}_{lex}, \mathbf{cells}_{lex}], \\ \mathbf{g}_f &= Softmax(Trans(\mathbf{t}_g)), \\ \mathbf{t}_{lex}[i] &= \sum_{i \in |V|} \sum_{j=1}^{|F|} \mathbf{g}_f[j][i] \cdot \mathbf{t}_f[j][i], \end{aligned} \tag{8}$$

where F represents the set of fields in the table, $|F|$ is the number of fields. $\mathbf{t}_g \in \mathcal{R}^{|F| \times h}$ is the list of hidden states of $[TTL]$, $[HEAD]$, and $[CELL]$, $\mathbf{t}_f \in \mathcal{R}^{|F| \times |V|}$ is the sparse representations of different fields stacked, $\mathbf{g}_f \in \mathcal{R}^{|F| \times |V|}$ adjusts each field of the inflow representation. The final sparse representation is obtained by a weighted aggregation across fields, where $\mathbf{g}_f[j][i]$ is the importance score of the token i in field j, $\mathbf{t}_f[j][i]$ is the lexical feature (e.g., occurrence probability) of the token i in field j.

We employ the inner product operation as the similarity function, consistent with semantic matching scores.

$$s_{lex}(q, t) = sim(\mathbf{q}_{lex}, \mathbf{t}_{lex}). \tag{9}$$

4.5 Hybrid Training

To enable retrievers to effectively learn both global semantic matching and field-aware lexical matching concurrently, we implement a dropout training strategy:

Matching Score Dropout. During training, we compute the final relevance score as either the semantic matching score $s_{sem}(q, t)$ with probability p_{sem} or the lexical matching score $s_{lex}(q, t)$ with probability p_{lex}. For the remaining training steps, we use the sum of them as the relevance score:

$$s(q,t) = \begin{cases} s_{sem}(q,t), & p_{sem}, \\ s_{lex}(q,t), & p_{lex}, \\ s_{sem}(q,t) + s_{lex}(q,t), & 1 - p_{sem} - p_{lex}. \end{cases} \tag{10}$$

This approach enables both independent and joint learning of global semantic matching and field-aware lexical matching, ensuring the development of each component while promoting their effective integration. In our experiments, we set $p_{sem} = p_{lex}$ since we consider them equally important for matching.

Loss Function. We adopt the InfoNCE loss for training (van den Oord et al., 2019). Specifically, for a query q_i in a batch, we pair a positive table t_i^+, with a set of random negative tables (positive tables from the other queries in the batch, e.g., $\{t_{i,j}^-\}$ for query q_j in the batch), the relevance loss for this sample is computed as:

$$\ell_{rel} = -log \frac{e^{s(q_i, t_i^+)}}{e^{s(q_i, t_i^+)} + \sum_j e^{s(q_i, t_{i,j}^-)}}. \tag{11}$$

We further employ the $FLOPS$ regularization (Paria et al., 2020) to constrain computational complexity during the training process, the training objective can be defined as follows:

$$\ell_{all} = \begin{cases} \ell_{rel} + (\lambda_q \ell_{FLOPS}^q + \lambda_t \ell_{FLOPS}^t), & s(q,t) = s_{lex}(q,t), \\ \ell_{rel}, & otherwise. \end{cases} \tag{12}$$

The regularization weights (λ_q and λ_t) enforce sparsity constraints for queries and tables on lexical matching, which is critical for fast retrieval.

During inference, we sum the semantic and lexical matching scores as the final relevance score:

$$s(q, t) = s_{sem}(q, t) + s_{lex}(q, t). \tag{13}$$

4.6 Efficiency Discussion

Let h be the dimension of the dense vectors yielded by PLMs and d be the number of non-zero items in sparse representations. During training, the computational costs of the different retrievers are as follows: single-vector dense retrievers like BIBERT require $O(|q|^2 h + |t|^2 h)$ for encoding, followed by $O(h^2)$ for matching, $|q|$ and $|t|$ are the lengths of the query and table sequence. Multi-vector dense retrievers such as SSDR$_{im}$ incur the same encoding cost of $O(|q|^2 h + |t|^2 h)$, but their matching costs scale to $O(\alpha\beta h^2)$, where α denotes the number of vectors used to represent a query and β is the number of vectors per tables. Similarly, the total computational cost of SPLADE is $O(|q|^2 h + |t|^2 h) + O(d^2)$. Although d exceeds h at initialization due to the large vocabulary size, it gradually approaches h under $FLOPS$ regularization. From the overall training process, SPLADE costs about as much as BIBERT. For THYME, the computational cost is $O(|q|^2h+|t|^2h)+O(h^2+d^2)$. This is slightly higher than that of BIBERT and SPLADE. But it remains significantly lower than that of multi-vector dense retrievers like SSDR$_{im}$.

During inference, since the dense and sparse representations are constructed independently, semantic matching and exact matching can be done in parallel. THYME does not introduce an additional time delay and is as efficient as its backbone models, BIBERT and SPLADE.

5 Experimental Settings

5.1 Datasets

We conduct experiments on two standard table retrieval benchmarks:

- **NQ-TABLES** (Herzig et al., 2021) is a subset of the Natural Questions (NQ) (Kwiatkowski et al., 2019), collected from Wikipedia and search engine logs.
- **OTT-QA** (Kostić et al., 2021) is an open-domain multi-hop QA dataset from Wikipedia, containing both textual and tabular corpus. We use the subset related to tables for retrieval evaluation.

The statistics of our benchmarks are shown in Table 1. We also show representative samples of these benchmarks in the Appendix B.

		NQ-TABLES		OTT-QA	
		Train	Test	Train	Test
Query	Count	9,594	919	41,469	2,214
	Avg. # Words.	8.94	8.90	21.79	22.82
Table	Count	169,898	169,898	419,183	419,183
	Avg. # Row.	10.70	10.70	12.90	12.90
	Avg. # Col.	6.10	6.10	4.80	4.80
# Relevant Tables per Query		1.00	1.05	1.00	1.00

Table 1: Statistics of benchmarks. Note that there are 919 unique queries and 966 query-table pairs in the test set of NQ-TABLES.

5.2 Baselines

We compare THYME with the following baselines. **Sparse Retrievers**: BM25 (Robertson and Walker, 1994) and SPLADE (Formal et al., 2021). **Dense Retrievers**: We selected three groups of dense retrievers as our baselines. (1) Single-vector text retrievers, such as BIBERT (Lin et al., 2021) and PRE-DPR (Wang et al., 2022), which had been trained on text retrieval corpus with relevance matching capability. (2) Single-vector table retrievers, such as TAPAS (Herzig et al., 2020) and DTR (Herzig et al., 2021). (3) Table retrievers that use multi vectors, such as SSDR$_{im}$ (Jin et al., 2023), which extracts multi vectors to represent both queries and tables. **Hybrid Retrievers**: We introduced hybrid retrievers such as DHR (Lin et al., 2023), along with two of our implementations: BIBERT-BM25$_{sf}$ to investigate the impact of score fusion and BIBERT-SPLADE$_{tf}$ to analyze the joint training of hybrid representations based on a shared encoder. Details of baselines are shown in the Appendix A.

Additionally, there are some methods, such as TaBERT (Yin et al., 2020) and StruBERT (Trabelsi et al., 2022), which are not used as our baselines, since they are designed for table reranking, not retrieval. There are also some LLM-based retriever (BehnamGhader et al., 2024; Lee et al., 2025) that achieve remarkable performance in text retrieval. Due to resource constraints, we do not choose these models as our baselines. THYME improves the performance of table retrieval from the perspective of field-aware hybrid matching. It is orthogonal to backbones' optimization and can be integrated into different backbones to yield cumulative performance improvements.

5.3 Evaluation Metrics

We use recall and normalized discounted cumulative gain (NDCG) for evaluation. We apply a cutoff at 50 for retrieved tables and report R@1, R@10, and R@50 to show how many relevant tables are retrieved, following Herzig et al. (2020) and Jin et al. (2023). Since high-ranking tables in retrieval results serve as the inputs for downstream tasks (e.g., table comprehension, tableQA, etc.), we use NDCG to evaluate whether relevant tables are ranked to top positions. Given that queries in our test set typically have only one annotated relevant table, NDCG@1 closely aligns with R@1 and NDCG@50 shows limited discriminative power due to minimal score variation. We select NDCG@5 and NDCG@10 as our primary evaluation metrics, as they provide practical relevance to real-world applications where only the top few results are examined. Statistical significance is measured with two-tailed t-tests with $p < 0.05$.

6 Results and Discussion

6.1 Overall Retrieval Performance

Table 2 shows the performance of three groups of baselines: sparse, dense, and hybrid retrievers, on

		NQ-TABLES					OTT-QA				
		NDCG@5	NDCG@10	R@1	R@10	R@50	NDCG@5	NDCG@10	R@1	R@10	R@50
Sparse	BM25	25.52	27.12	18.49	36.94	52.61	35.09	37.45	23.98	51.94	69.11
	SPLADE	53.70	56.75	39.84	83.33	94.65	75.27	76.72	62.74	89.52	95.21
Dense	BIBERT	60.49	63.16	43.78	82.25	93.71	70.57	72.49	56.82	86.50	94.26
	PRE-DPR	63.05	66.13	45.32	85.84	95.44	67.92	70.00	53.43	85.95	93.22
	TAPAS*	61.77	64.29	43.79	83.49	95.10	70.89	72.72	57.86	86.77	94.04
	DTR*	51.04	53.98	32.62	75.86	89.77	56.68	58.94	42.10	75.75	88.80
	SSDR$_{im}$	62.31	65.02	45.47	84.00	95.05	69.81	71.76	56.96	86.22	93.55
Hybrid	DHR	61.16	64.32	43.67	84.65	95.62	75.27	76.65	63.64	88.48	95.30
	BIBERT-BM25$_{sf}$	53.81	57.19	35.87	79.63	94.56	71.36	73.29	59.49	86.81	94.67
	BIBERT-SPLADE$_{tf}$	63.24	66.25	45.62	**86.72**	95.62	76.90	78.30	64.72	91.01	**96.34**
	THYME	**65.72**†	**68.14**†	**48.55**†	86.38	**96.08**	**78.21**†	**79.58**†	**66.67**†	**91.10**	96.16

Table 2: Overall table retrieval performance. **Bold** and underline indicate the best and suboptimal performance respectively. We set the batch size to 144 for all methods, and the difference with the corresponding paper is denoted by $\star$. Statistically significant (p < 0.05) improvements over BIBERT-SPLADE$_{tf}$ are marked with †.

NQ-TABLES and OTT-QA. It shows that hybrid retrievers perform better than dense and sparse retrievers. Even the simple combination of BIBERT and SPLADE boosts the performance by a large margin. Among all the methods, THYME performs the best, significantly better than the SOTA baselines, indicating its efficacy in conducting field-aware hybrid matching.

We also have the following observations: (1) Dense retrievers excel on NQ-TABLES, while sparse retrievers perform better on OTT-QA. The reason for this disparity is that queries in OTT-QA are obtained by decontextualizing questions from the closed-domain QA dataset, which contains more detailed information compared with queries in NQ-TABLES. Hybrid retrieval bridges this gap through combined semantic and exact matching, demonstrating robust performance across diverse queries. THYME takes it a step further. It calibrates the retrieval preferences of various fields to make it a compelling solution for table retrieval. (2) THYME utilizes field indicator tokens in table inputs to facilitate adaptive differentiation and aggregation of information across different fields. BIBERT achieves comparable results to TAPAS using the same approach. This suggests that the model can adaptively learn the structure of the table, and neural models designed for tables may be not necessary for retrieval. (3) PRE-DPR, trained for text retrieval, also shows competitive performance in table retrieval, which suggests that relevance learned from text matching also benefits table retrieval.

6.2 Analyses on Model Variants

To derive the sparse representations of the table, we perform table-structure pooling and aggregate these representations using MoFE. To evaluate the impact of our design, we train and evaluate alternative variants for both components. For pooling within the field, we also tried max or mean pooling on all the tokens instead of the table-structure pooling in THYME. For the aggregation over fields, we attempted max and mean pooling. Notably, using max/mean pooling both within and across fields degrades to treating tables as unstructured text and representing them with SPLADE. Table 3 shows how the variants of THYME perform with the revised sparse representations.

Pooling	Aggregation	NDCG@5	NDCG@10	R@10
Table-Structure	MoFE	**65.72**	**68.14**	**86.38**
Max	MoFE	62.18*	64.52*	85.79
Mean	MoFE	60.61*	63.96*	85.09*
Max	Max	61.44*	64.19*	85.42
Mean	Mean	57.72*	60.96*	83.15*

Table 3: Comparisons of pooling and aggregation methods for sparse representations on NQ-TABLES. '$\star$' indicates statistically significant differences (p<0.05) with THYME (the first row).

We can see that (1) table structure in the sparse representation can not be ignored; (2) max pooling has better performance than mean pooling in terms of sparse field representations, consistent with the observations from SPLADE on text retrieval (Formal et al., 2021), but both are significantly worse than our table-structure pooling approach and (3) MoFE is better than using max or mean pooling to aggregate the field representations, indicating field importance in its final sparse representations are better learned.

6.3 Matching Preferences of Different Fields

To see whether table titles and bodies have different relevance matching preferences, we compare THYME variants that selectively mask dense/sparse representations of titles or bodies. For instance, when only dense representations of titles are used, their sparse representations are not aggregated to the tables' final sparse representations. Table 4 shows the performance of these variants. We can observe: (1) The absence of title sparse representations results in a smaller degradation compared to other variants; (2) Performance declines more significantly when sparse representations of table bodies are omitted than when dense representations are removed; (3) THYME exhibits the worst performance when only sparse title representations and dense body representations are used. The comparison of different variants reveals that titles benefit primarily from semantic matching, whereas table bodies (headers and cells) depend more critically on lexical matching.

Title	Headers&Cells	NDCG@5	NDCG@10	R@10
Dense, Sparse	Dense, Sparse	**65.72**	**68.14**	**86.83**
Dense	Dense, Sparse	63.97*	67.18	86.72
Dense, Sparse	Sparse	63.81*	66.48*	85.65*
Dense, Sparse	Dense	57.98*	61.14*	82.48*
Sparse	Dense	57.48*	60.25*	82.15*

Table 4: Study on the impact of representation types of table fields on NQ-TABLES. '⋆' marks the statistically significant difference ($p<0.05$) compared to THYME (the first row).

7 Application in TableQA

In the era of LLMs, retrievers are often used as a component of a retrieval-augmented generation (RAG) system to provide context for LLMs. We evaluated THYME's practical value in Open Domain TableQA via RAG using Mistral (Jiang et al., 2023), Llama3 (Grattafiori et al., 2024), and Qwen2.5 (Qwen et al., 2025). NQ-TABLES contains factual questions, relevant tables, and corresponding answers. We measure the accuracy of whether the LLMs' outputs contain the ground-truth answers. Results are shown in Table 5.

When more results are used for augmentation, QA performance becomes better as well. Due to incomplete annotation of the relevant tables in the NQ-TABLES, the tables retrieved by the model that are not labeled as relevant may contain information that is related to the query. This requires the retriever to learn the relevance matching between queries and tables, rather than fitting the data. Among existing table retrievers, THYME demonstrates the best RAG performance.

Retriever	LLM	Accuracy n=1	Accuracy n=3	Accuracy n=5
SPLADE	Mistral-7B	29.61	35.42	34.95
	Llama3-8B	32.07	37.17	33.88
	Qwen2.5-7B	31.90	35.79	37.35
BIBERT	Mistral-7B	32.93	33.46	35.30
	Llama3-8B	32.40	33.03	34.17
	Qwen2.5-7B	34.80	37.92	37.01
SSDR$_{im}$	Mistral-7B	32.76	36.95	36.30
	Llama3-8B	34.25	38.14	37.89
	Qwen2.5-7B	33.42	39.27	39.66
BIBERT-SPLADE$_{tf}$	Mistral-7B	32.67	33.59	35.71
	Llama3-8B	32.66	34.67	34.55
	Qwen2.5-7B	33.24	36.76	37.30
THYME	Mistral-7B	35.48	37.59	37.20
	Llama3-8B	36.14	39.16	39.29
	Qwen2.5-7B	**37.28**	**40.28**	**41.20**

Table 5: End-to-end tableQA performance. **Bold** indicates the best performance.

8 Case Study

Figure 3 compares THYME with SOTA baselines on a query about the 2018 Olympics opening ceremony time. Only THYME ranks the relevant table at the top. "2018", "Olympics", "opening", and "ceremony" occur in the titles and/or cells while "when" is semantically matched with "Date" and "Time" in the relevant table. In contrast, the strongest baselines BIBERT-SPLADE$_{tf}$, and SSDR$_{im}$ prioritize exact keyword matching while neglecting the semantic matching to "when for 2018" that needs date or time to answer. It shows that THYME has successfully learned field-aware hybrid matching.

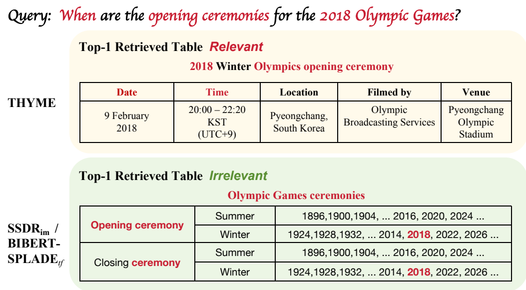

Figure 3: Top-1 retrieved table from different retrievers.

9 Conclusion

In this work, we propose a retriever based on the observation that table cells could prefer local matching of detailed information which differs from unstructured text such as table titles. We tailor the representations for tables and incorporate both dense and sparse representations to better suit the matching needs at different granularities. Experimental results show that our proposed method can adaptively balance the semantic and lexical matching requirements among the table fields.

Limitations

This paper investigates the preferences in matching across different fields in a table during the retrieval. Tables represent a critical category of structured data that coexists with other prevalent formats (e.g., HTML, PDF) in real-world information systems. Our method demonstrates effectiveness for table-structured data. How to extend it to a wider range of data formats needs to be further explored.

Ethics Statement

We approach ethics with great care. In this paper, all the datasets we use are open-source, which are widely adopted in previous research. These datasets are collected from publicly available Internet such as Wikipedia. The methods covered in the paper with their checkpoints, are also from the open-source community. There are no ethics-related issues involved.

References

Yang Bai, Xiaoguang Li, Gang Wang, Chaoliang Zhang, Lifeng Shang, Jun Xu, Zhaowei Wang, Fangshan Wang, and Qun Liu. 2020. Sparterm: Learning term-based sparse representation for fast text retrieval. *Preprint*, arXiv:2010.00768.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. Ms marco: A human generated machine reading comprehension dataset. *Preprint*, arXiv:1611.09268.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. Llm2vec: Large language models are secretly powerful text encoders. *Preprint*, arXiv:2404.05961.

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*, IDEA '13, page 18–26, New York, NY, USA. Association for Computing Machinery.

Michael J Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. Webtables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment*, 1(1):538–549.

Nuo Chen, Linjun Shou, Ming Gong, Jian Pei, Chenyu You, Jianhui Chang, Daxin Jiang, and Jia Li. 2023. Bridge the gap between language models and tabular understanding. *Preprint*, arXiv:2302.09302.

Wenhu Chen, Ming-Wei Chang, Eva Schlinger, William Wang, and WilliamW. Cohen. 2020a. Open question answering over tables and text. *arXiv: Computation and Language,arXiv: Computation and Language*.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020b. Tabfact: A large-scale dataset for table-based fact verification. *Preprint*, arXiv:1909.02164.

Xilun Chen, Kushal Lakhotia, Barlas Oguz, Anchit Gupta, Patrick S. H. Lewis, Stan Peshterliev, Yashar Mehdad, Sonal Gupta, and Wen-tau Yih. 2021. Salient phrase aware dense retrieval: Can a dense retriever imitate a sparse one? *CoRR*, abs/2110.06918.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the trec 2019 deep learning track. *Preprint*, arXiv:2003.07820.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. Splade v2: Sparse lexical and expansion model for information retrieval. *Preprint*, arXiv:2109.10086.

Luyu Gao, Zhuyun Dai, Zhen Fan, and Jamie Callan. 2020. Complementing lexical retrieval with semantic residual embedding. *CoRR*, abs/2004.13969.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, and et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Jiafeng Guo, Yinqiong Cai, Keping Bi, Yixing Fan, Wei Chen, Ruqing Zhang, and Xueqi Cheng. 2025. Came: Competitively learning a mixture-of-experts model for first-stage retrieval. *ACM Transactions on Information Systems*, 43(2):1–25.

Jonathan Herzig, Thomas Müller, Syrine Krichene, and Julian Martin Eisenschlos. 2021. Open domain question answering over tables via dense retrieval. *arXiv preprint arXiv:2103.12011*.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Sujay Kumar Jauhar, Peter Turney, and Eduard Hovy. 2016. Tables as semi-structured knowledge for question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 474–483.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Nengzheng Jin, Dongfang Li, Junying Chen, Joanna Siebert, and Qingcai Chen. 2023. Enhancing open-domain table question answering via syntax- and structure-aware dense retrieval. *Preprint*, arXiv:2309.10506.

Bogdan Kostić, Julian Risch, and Timo Möller. 2021. Multi-modal retrieval of tables and texts using tri-encoder models. *Preprint*, arXiv:2108.04049.

Saar Kuzi, Mingyang Zhang, Cheng Li, Michael Bendersky, and Marc Najork. 2020. Leveraging semantic and lexical matching to improve the recall of document retrieval systems: A hybrid approach. *CoRR*, abs/2010.01195.

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

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2025. Nv-embed: Improved techniques for training llms as generalist embedding models. *Preprint*, arXiv:2405.17428.

Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2021. *Pretrained Transformers for Text Ranking: BERT and Beyond*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Sheng-Chieh Lin, Minghan Li, and Jimmy Lin. 2023. Aggretriever: A simple approach to aggregate textual representations for robust dense passage retrieval. *Preprint*, arXiv:2208.00511.

Sheng-Chieh Lin and Jimmy Lin. 2021. Densifying sparse representations for passage retrieval by representational slicing. *CoRR*, abs/2112.04666.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *Preprint*, arXiv:1907.11692.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2020. Sparse, dense, and attentional representations for text retrieval. *CoRR*, abs/2005.00181.

Biswajit Paria, Chih-Kuan Yeh, Ian E. H. Yen, Ning Xu, Pradeep Ravikumar, and Barnabás Póczos. 2020. Minimizing flops to learn efficient sparse representations. *Preprint*, arXiv:2004.05665.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, organised by Dublin City University*, pages 232–241. Springer.

Tao Shen, Xiubo Geng, Chongyang Tao, Can Xu, Guodong Long, Kai Zhang, and Daxin Jiang. 2023. Unifier: A unified retriever for large-scale retrieval. *Preprint*, arXiv:2205.11194.

Roee Shraga, Haggai Roitman, Guy Feigenblat, and Mustafa Cannim. 2020. Web table retrieval using multimodal deep learning. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 1399–1408, New York, NY, USA. Association for Computing Machinery.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.

Mohamed Trabelsi, Zhiyu Chen, Shuo Zhang, Brian D. Davison, and Jeff Heflin. 2022. Strubert: Structure-aware bert for table search and matching. In *Proceedings of the ACM Web Conference 2022*, WWW '22. ACM.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2019. Representation learning with contrastive predictive coding. *Preprint*, arXiv:1807.03748.

Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. 2021. Tuta: Tree-based transformers for generally structured table pre-training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21. ACM.

Zhiruo Wang, Zhengbao Jiang, Eric Nyberg, and Graham Neubig. 2022. Table retrieval may not necessitate table-specific model design. In *Proceedings of the Workshop on Structured and Unstructured Knowledge Integration (SUKI)*, pages 36–46, Seattle, USA. Association for Computational Linguistics.

Pengcheng Yin, Graham Neubig, Wen tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. *Preprint*, arXiv:2005.08314.

Shuo Zhang and Krisztian Balog. 2018. Ad hoc table retrieval using semantic similarity. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*, WWW '18. ACM Press.

Shuo Zhang and Krisztian Balog. 2020. Web table extraction, retrieval, and augmentation: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(2):1–35.

A Details of Baselines

Sparse Retrievers:

- **BM25 (Robertson and Walker, 1994)** is a well-known retrieval method that estimates the relevance of documents to a user query based on bag-of-words representations and exact term matching.

- **SPLADE (Formal et al., 2021)** is a sparse retriever based on BERT and one of the backbones of our model. It maps a query or document to a vector of the vocabulary size, where each dimension corresponds to the probability of a term.

Dense Retrievers:

- **BIBERT (Lin et al., 2021)** is a standard dense retriever based on BERT. It is also one of the backbones of our model. The hidden state of [CLS] for a query and a document from BERT is used to estimate the relevance score.

- **PRE-DPR (Wang et al., 2022)** is a text retriever that has been fine-tuned with a large text corpus.

- **TAPAS (Herzig et al., 2020)** utilizes distinct types of embeddings like row and column embeddings to represent structure. It is also pre-trained on a large amount of tabular data and fine-tuned on cell, row, and column-level tasks. It is a universal table encoder that is widely used in table-related tasks.

- **DTR (Herzig et al., 2021)** uses TAPAS as the encoder and has been fine-tuned with relevant data of tables and queries.

- **SSDR$_{im}$ (Jin et al., 2023)** is the state-of-the-art (SOTA) table retriever, which extracts the vectors of nouns to represent the query. For tables, it constructs representations of rows and columns by pooling, a part of which is sampled as the representation of tables.

Hybrid Retrievers:

- **BIBERT-BM25$_{sf}$** is a hybrid retrieval method that obtains the relevance score by directly adding the scores of semantic matching based on BIBERT and exact matching from BM25.

- **BIBERT-SPLADE$_{tf}$** is a simple fusion of BIBERT and SPLADE. The outputs of both are used to estimate semantic matching and lexical matching respectively. Similar to BIBERT-BM25$_{sf}$, the relevance score comes from the sum of the semantic matching score and lexical matching score.

- **DHR (Lin et al., 2023)** densifies the sparse representation and concatenates it with the dense representation to construct a single representation. It is compatible with most retrieval frameworks.

B Case Overview of Benchmarks

To effectively visualize and compare the differences in queries between NQ-TABLES and OTT-QA, we show samples from each of them in Figure 4 and Figure 5.

Query: *What is the elevation of eagle creek oregon?*

Relevant Table:

Eagle Creek (Oregon)

Name	Type	Elevation	Coordinate	USGS Map	GNIS ID
West Eagle Creek (Union County, Oregon)	Stream	4,426 ft (1,349 m)	45 ° 01′10″N 117 ° 27′15″W	Bennet Peak	1128826
Eagle Butte Creek (Lane County, Oregon)	Stream	1,765 ft (538 m)	43 ° 47′34″N 122 ° 19′24″W	Huckleberry Mountain	1141461
Eagle Creek, Oregon	Populated Place	344 ft (105 m)	45 ° 21′26″N 122 ° 21′32″W	Estacada	1120258
...					
West Eagle Creek (Union County, Oregon)	Stream	4,426 ft (1,349 m)	45 ° 01′10″N 117 ° 27′15″W	Bennet Peak	1128826

Website: https://en.wikipedia.org//w/index.php?title=Eagle_Creek_(Oregon)&oldid=738892320

Answer: *344 ft (105 m)*

Figure 4: A case of NQ-TABLES.

Query: *Which male athlete was born in Alabama and had a 400 meter time under 44.5 seconds?*

Relevant Table:

400 Metres

Year	Time	Role	Place
1966	44.82y	Wendell Mottley (TTO)	Kingston
1967	44.74+h	Tommie Smith (USA)	San Jose
...			
1984	44.27	Alonzo Babers (USA)	Los Angeles
...			
2024	43.40	Quincy Hall (USA)	Saint-Denis

Website: https://en.wikipedia.org/wiki/400_metres

Answer: *Alonzo Babers*

Figure 5: A case of OTT-QA.

C Implementation Details

We initialize THYME, BIBERT, and SPLADE with BERT-base. For the other baselines, we use the released checkpoints for initialization. To ensure a fair comparison, we maintain identical batch size, learning rate, and training steps across all trained models. With a batch size of 144 and a learning rate of $1e-5$, we compare the performance of the different models after 50 training epochs. For THYME, we set $p_{sem} = p_{lex} = 0.15$ for matching score dropout and $\lambda_q = \lambda_t = 1e-4$ for $FLOPS$ regularization.

D Prompt for TableQA

The prompt we used in evaluating the effect of different table retrievers on the answers generated by LLM is shown in Figure 6.

System: You are a helpful assistant.
User: Answer the question based on the table provided, outputting the answer directly, not the reasoning process or other additional information.
<Question>: Who is the owner of reading football club?
<Tables>: [Table 1]: Reading F.C., ['Full name', 'Nickname(s)', 'Founded', 'Ground', 'Capacity', 'Owner', 'Chairman', 'Manager', 'League', '2016–17', 'Website', '', '', 'Home colours'], [['Reading Football Club', 'The Royals', '1871; 147 years ago', 'Madejski Stadium', '24,161[1]', 'Dai Yongge and Dai Xiuli (majority)', 'Sir John Madejski', 'Jaap Stam', 'Championship', 'Championship, 3rd', 'Club website', '', 'Home colours Away colours', 'Away colours']].
<Answer>:

Assistant: Dai Yongge and Dai Xiuli

Figure 6: The prompt for tableQA.

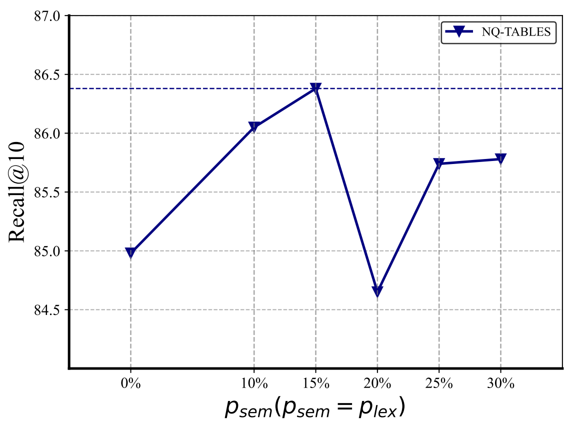

Figure 7: Impact of dropout rate.

E Hyper-parameter Sensitivity

Dropout Rate. During training, we introduce a dropout strategy. To see how the dropout ratio p_{sem} and p_{lex} (note that we set $p_{sem} = p_{lex}$) impact the retrieval performance of THYME, we vary the probability from 0% to 30% and examine how Recall@10 changes. From Figure 7, we can see the performance fluctuates on both datasets when the ratio is set to larger values. However, the best performance is always achieved when p_{sem} and p_{lex} are above 0, which means our matching score dropout strategy is beneficial.