HELIOS: Harmonizing Early Fusion, Late Fusion, and LLM Reasoning for Multi-Granular Table-Text Retrieval

Anonymous ACL submission

Abstract

Table-text retrieval aims to retrieve relevant tables and text to support open-domain question answering. Existing studies use either early or late fusion, but face limitations. Early fusion pre-aligns a table row with its associated passages, forming "stars," which often include irrelevant contexts and miss query-dependent relationships. Late fusion retrieves individual nodes, dynamically aligning them, but it risks missing relevant contexts. Both approaches also struggle with advanced reasoning tasks, such as column-wise aggregation and multi-hop reasoning. To address these issues, we propose HELIOS, which combines the strengths of both approaches. First, the *edge-based bipartite subgraph retrieval* identifies finer-grained edges between table segments and passages, effectively avoiding the inclusion of irrelevant contexts. Then, the *query-relevant node expansion* identifies the most promising nodes, dynamically retrieving relevant edges to grow the bipartite subgraph, minimizing the risk of missing important contexts. Lastly, the *star-based LLM refinement* performs logical inference at the star graph level rather than the bipartite subgraph, supporting advanced reasoning tasks. Experimental results show that HELIOS outperforms state-of-the-art models with a significant improvement up to 42.6% and 39.9% in recall and nDCG, respectively, on the OTT-QA benchmark.

1 Introduction

Open-domain question answering (ODQA) over tables and text is important as it leverages the complementary strengths of structured and unstructured data. Tables offer vast amounts of related facts but lack diversity, while text provides broader contextual information (Chen et al., 2020b,a), making the integration of both modalities essential. Table-text retrieval plays a crucial role in ODQA by retrieving relevant tables and text to support retriever-reader systems (Chen et al., 2020a; Ma et al., 2023, 2022).

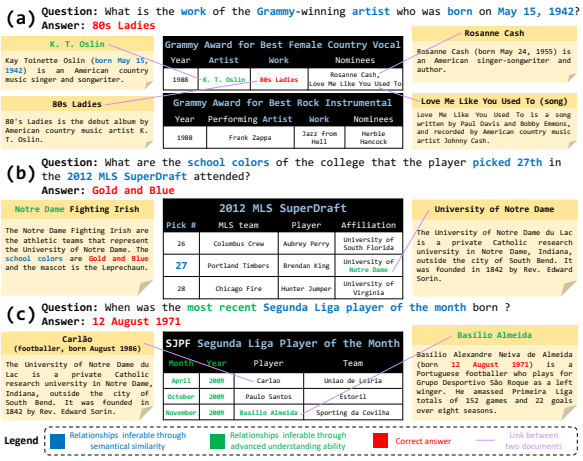


Figure 1: Simplified examples of three cases where existing methods struggle to retrieve the question-related documents correctly. (a) Inadequate granularity of retrieval units leading to inaccurate retrieval results. (b) Entity linking results cannot estimate essential query-aware relationships. (c) Inability of advanced reasoning such as table aggregation and multi-hop reasoning.

Despite its importance, table-text retrieval is challenging due to the need to bridge structured tables and unstructured passages. Tables encode information in rows and columns, requiring structural understanding, while passages follow a narrative format. Effective retrieval demands resolving multi-hop relationships across these distinct formats.

Existing methods have achieved some success by employing either *early* or *late fusion* techniques in their top-k retrieval. The *early fusion* attempts to reduce the search space by grouping relevant documents before a query is presented. It pre-aligns a table row with associated passages via entity linking, creating a *fused block* as the retrieval unit (Chen et al., 2020a; Huang et al., 2022; Kang et al., 2024). In contrast, the late fusion aligns relevant table rows and passages dynamically using query-based similarity matching after the query is given. It returns a ranked sequence of evidence chains, where an *evidence chain* refers to a pair consisting of a table

row and a passage (Ma et al., 2022, 2023).

However, the existing studies have several significant limitations.

(1) Inadequate granularity of retrieval unit. Early fusion strategy constructs retrieval units independently of the query, often including query-irrelevant passages, which distorts similarity calculations between fused blocks and queries. For example in Figure 1(a), entity linking connects the `Grammy Award for Best Female Country Vocal` table with four surrounding passages, even though only the information on the `K. T. Oslin` is relevant (Figure 1(a)). In the late fusion strategy, retrieving a single table segment or passage may be partially relevant to a query, incurring the risk of retrieving incorrect tables. For instance, during the first iteration of retrieval, the system might retrieve the `Grammy Award for Best Rock Instrumental` table instead of the correct one. Both tables share overlapping terms such as `Grammy`, `Artist`, and `Work`, causing ambiguity in target identification.

(2) Missing query-dependent relationships. The early fusion strategy relies on entity linking to predefine relationships between tables and passages, failing to capture query-dependent links that might contain the information necessary to answer the query. For instance, in Figure 1(b), the table `2012 MLS SuperDraft` is early fused with the entity `University of Notre Dame`. However, when the question specifies the information about `school colors`, it should be linked to the `Notre Dame Fighting Irish` passage.

(3) Lack of advanced reasoning. Queries that require complex reasoning, such as multi-hop or column-wise aggregation, often demand advanced logical inference beyond simple semantic similarity with the query. Since previous approaches rely on semantic similarity, they might fail to retrieve rows or passages identifiable through logical inference. For example, in Figure 1(c), the query involves understanding the `most recent Segunda Liga Player of the Month` is `Basilio Almeida`, where the row with the latest `Year` and `Month` combination has to be inferred.

To systematically address these limitations, we first formalize the terms proposed in previous studies using a *bipartite graph*, where table segments and passages are represented as two disjoint sets of nodes, and the links between them are represented as edges. Therefore, the term *fused block* used in the early fusion strategy (Chen et al., 2020a; Huang et al., 2022; Kang et al., 2024) can be represented as a star (Diestel, 2024) centered on a node of type `table segment`, with connected nodes of type `passage`. Similarly, the *evidence chain* used in the late fusion strategy (Ma et al., 2022, 2023) corresponds to an edge connecting a pair of nodes: one of type `table segment` and one of type `passage`.

Based on this formalization, we propose `HELIOS`, a novel graph-based retrieval consisting of three stages: early fusion, late fusion, and LLM reasoning. Specifically, `HELIOS` adopts the following three key ideas:

(1) Combined usage of early and late fusion. We selectively leverage the advantages of both early fusion and late fusion. Early fusion pre-aligns tables with related passages to mitigate the risk of retrieving incomplete or partially relevant information inherent in late fusion, while late fusion dynamically resolves document relationships to address early fusion's reliance on predefined links.

(2) Graph refinement. We leverage LLMs to perform further advanced reasoning over the retrieved graph, enabling deeper logical inference beyond simple semantic similarity. For instance, in Figure 1(c), when the `SJPF Segunda Liga Player of the Month` table is retrieved, the LLM can perform aggregation to identify the most recent player and conduct multi-hop reasoning to select the corresponding passage for `Basilio Almeida`.

(3) Granularity determination for each retrieval stage. In our retrieval pipeline, each stage early fusion, late fusion, and graph refinement serves a distinct purpose, necessitating the precise determination of the appropriate operational units for each. For the early fusion stage, we propose an edge-level, multi-vector-based retrieval, striking a balance between eliminating irrelevant contexts in star graph retrieval and avoiding the partial information problem in node-based retrieval. In the late fusion stage, we set the unit as an individual node. We identify query-relevant nodes within the graph produced by the early fusion stage so that we can design the late fusion process to expand the graph using only nodes closely aligned with the query context. This approach mitigates the challenge where the earlier stage may retrieve nodes irrelevant to the query. Finally, the graph refinement stage presents an expanded graph from late fusion to the LLM, reducing hallucination risks by decomposing the graph into smaller star graphs.

We evaluate competitors and `HELIOS` on two datasets. Experimental results demonstrate that `HELIOS` significantly outperforms SOTA systems.

2 Related Work

2.1 Open-domain Question Answering

Open-Domain Question Answering (ODQA) aims to answer factual questions using a large knowledge corpus (Zhang et al., 2023). Standard benchmarks like Natural Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and SearchQA (Dunn et al., 2017) focus on single-hop queries, where answers reside within a single passage of unstructured text. Further advances were shown by HotpotQA (Yang et al., 2018) and WikiHop (Welbl et al., 2018), presenting challenging queries that require multi-hop reasoning across multiple passages. However, these benchmarks consider only unstructured text and do not address multi-hop reasoning over both structured tables and unstructured passages, which is essential in table-text retrieval tasks. OTT-QA (Chen et al., 2020a) is the first ODQA benchmark designed to support multi-hop reasoning between tables and text, requiring retrieval and reasoning across both modalities.

2.2 Table-Text Retrieval

Table-text retrieval methods can be categorized into early fusion and late fusion approaches. These terms, originally used in multi-modal tasks such as image-sentence retrieval, describe whether different modalities are encoded jointly or separately (Wang et al., 2022; Snoek et al., 2005; Gadzicki et al., 2020). Similarly, in table-text retrieval, early fusion and late fusion approaches differ in whether tables and text are linked before or after the retrieval process (Kang et al., 2024).

Early fusion approaches (Chen et al., 2020a; Huang et al., 2022; Kang et al., 2024) pre-link table rows with associated passages via entity linking, forming fused blocks as retrieval units. While convenient, this approach has two drawbacks: (i) Fused blocks often include query-irrelevant passages, causing noisy retrieval and information loss in block-level embedding. (ii) Offline pre-linked blocks cannot adapt to query-dependent relationships discovered during retrieval (Figure 1(b)). To address (i), we adopt edges—a finer-grained retrieval unit—and employ late interaction during retrieval to minimize information loss and avoid large, noisy blocks. For (ii), we propose query-relevant node expansion to incorporate relationships that emerge in a specific query's context.

Late fusion approaches (Ma et al., 2022, 2023) dynamically form table-passage connections online. Although more flexible, they must consider many table-passage combinations, typically handled by beam search, which can cause error propagation. Our approach mitigates this using edge-based late interaction retrieval, which captures richer contextual relationships by pre-linking table segments with passages offline, enabling more accurate seed document retrieval.

Both early and late fusion approaches rely primarily on semantic similarity for retrieval, limiting their ability to retrieve table segments and passages requiring advanced reasoning (e.g., column-wise aggregation, multi-hop inference), as shown in Figure 1(c). To address this, we propose a star-based LLM refinement step, leveraging LLMs for logical inference to refine retrieval results.

Additionally, our retrieval method applies different levels of granularity (e.g., edges, nodes, stars) tailored to each retrieval phase. While DRAMA (Yuan et al., 2024) also employs a multi-granularity approach, it operates in a constrained setting where relevant tables and passages are provided, unlike our open-domain retrieval setting. GTR (Wang et al., 2021) and MGNETS (Chen et al., 2021) improve table encoding using graph-based methods, addressing a problem orthogonal to our focus. In contrast, our work centers on retrieving both tables and text upon their semantic relationships.

3 Preliminaries

3.1 Problem Formulation

Table-text retrieval is involved from a retrieval *corpus* $\mathcal{C}$, which comprises two distinct sets: a collection of passages $\mathcal{C}_P = \{P^{(1)}, \ldots, P^{(n)}\}$ and a collection of tables $\mathcal{C}_T = \{T^{(1)}, \ldots, T^{(m)}\}$. A *passage* is defined as a sequence of tokens P, representing unstructured text. A *table* is a structured matrix T, consisting of cells $T_{i,j}$, where i and j indicate the row index and the column index, respectively. Each cell $T_{i,j}$ may contain a number, date, phrase, or sentence. We define a document as either a passage or a table. Given a query q, the objective of table-text retrieval is to retrieve from corpus $\mathcal{C}$ a ranked list of documents such that the document containing the answer span a is positioned among the top results.

We split a table into multiple table segments, as commonly used in existing studies. Since a single table can easily exceed the token limits of language models, a table T is combined with its header to form a list of table segments $T = [S^{(1)}, \ldots, S^{(m')}]$ (Chen et al. 2020a). This process results in (i) a corpus $\mathcal{C}$ composed of table segments $\mathcal{C}_S$ and pas-

sages $\mathcal{C}_P$ (i.e., $\mathcal{C} = \mathcal{C}_S \cup \mathcal{C}_P$) and (ii) a mapping $\mathcal{M} : \mathcal{C}_S \rightarrow \mathcal{C}_T$ to associate table segments with their original table.

3.2 Table-Text Retrieval as Graph Retrieval

We adopt a graph representation, denoted by $G = (V, E, \Phi, \Gamma, \Lambda)$, to generalize various methods used in existing studies. Here, V is the set of vertices corresponding to a table segment or a passage, and E is the set of edges representing relationships between (table segment, passage) pairs. The mapping $\Phi : V \rightarrow \{\texttt{table segment}, \texttt{passage}\}$ maps each node to its type, while Γ maps a node to its attributes, such as the text of a passage or the matrix of table structures. The mapping $\Lambda : E \rightarrow \mathbb{R}$ maps each edge to its score.

The corpus can be expressed as the initial graph $G_{init} = (V_{init}, \emptyset, \Phi, \Gamma, \Lambda_{init})$, where each node in V_{init} one-to-one corresponds to a table segment or passage in $\mathcal{C}$. Early fusion generates table-text relationships via entity linking and updates G_{init} before a query q is presented. Given q, late fusion dynamically generates query-dependent table-text relationships to update G_{init}. Finally, we retrieve a query-relevant edge-scored bipartite graph $G_q = (V_q, E_q, \Phi, \Gamma, \Lambda_q)$ from G_{init}. This problem is often interpreted as ranking edges $\mathcal{E}_q$ from all possible edges, as retrieved results are fed to a reader with limited context size (Ma et al., 2022, 2023). $\mathcal{E}_q$ is typically obtained by sorting each edge e in G using its edge scores $\Lambda_q(e)$.

4 Proposed Method

We propose HELIOS, a novel graph-based retrieval framework that combines the strengths of early fusion, late fusion, and LLM reasoning. As shown in Figure 2, it operates in three stages: (i) **Edge-based Bipartite Subgraph Retrieval** retrieves edges from a bipartite data graph constructed via early fusion and integrates them into a single bipartite subgraph. (ii) **Query-relevant Node Expansion** enhances the retrieved subgraph by incorporating additional nodes through further retrieval. (iii) **Star-based LLM Refinement** refines the expanded graph through aggregation and multi-hop reasoning using the LLM.

4.1 Edge-based Bipartite Subgraph Retrieval

HELIOS initiates its process with the retrieval of a bipartite subgraph through two key steps: *early fusion* and *edge retrieval*.

(1) Early fusion: This step is performed offline, before a query is given. A bipartite data graph G_d is constructed by generating edges from G_{init}, which initially has no edges. Edge generation is composed of entity recognition and entity linking, following prior methods (Ma et al., 2022, 2023). Edges are generated between passage nodes and table segment nodes, resulting the generated data graph G_d to be a bipartite graph. G_d is then indexed in an edge-wise manner. Each edge $e = (S, P)$ is first linearized into a token sequence x.

$$x = [\ Linearize(\Gamma(S));\ \Gamma(P)\ ] \tag{1}$$

Then, x is embedded into a sequence of vectors.

$$\mathbf{X} = f_e(x) \in \mathbb{R}^{l_x \times d} \tag{2}$$

where l_x represents the length of a linearized edge x. We adopt the late-interaction-based multi-vector representation encoder ColBERTv2 (Santhanam et al., 2021) for f_e to create fine-grained embeddings. That is, HELIOS embeds edges, a larger unit compared to the node-level embeddings used in previous methods (Ma et al., 2022, 2023). Since edges contain more tokens than nodes, fine-grained embeddings are essential to reduce information loss.

(2) Edge Retrieval: When the query is given, we first identify the query-relevant edges by leveraging the semantic similarity between the query and edge embeddings.

$$\mathbf{Q} = f_e(q) \in R^{l_q \times d} \tag{3}$$

The similarity is then calculated between the query and each indexed edge.

$$sim(q, e; f_e) = \sum_{i=1}^{l_q} \max_{j \in [1, l_x]} \mathbf{Q}_i \mathbf{X}_j^T \tag{4}$$

We then select the top-k_1 edges that show the highest score $sim(q, e)$, measuring query-edge alignment. The selected edges are further passed to a fine-tuned all-to-all interaction reranker g_e, which performs a more detailed similarity evaluation. This identifies the most contextually relevant edges, allowing us to identify the top-k_2 query-relevant edges ($k_2 < k_1$). The fine-tuning process for f_e and g_e, including training dataset construction, is detailed in Appendix § A.1.

The resultant edges are integrated into the bipartite subgraph $G_c = (V_c, E_c, \Phi, \Gamma, \Lambda_c)$, Specifically, if there are duplicate nodes contained in the retrieved edges, those nodes are merged to form a single graph. G_c serves as the candidate bipartite subgraph, which becomes the foundation for further expansion and refinement. We save the similarity score for each edge in the score mapping function Λ_c, which is later used to sort the edges based on their relevance to the query in Section 4.3.

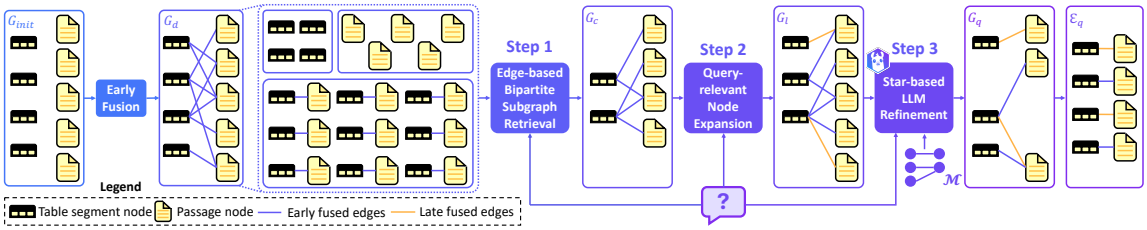


Figure 2: Overview of HELIOS: The initial graph G_{init} is early-fused to generate a graph G_d. Each node and edge of G_d are embedded. (1) The edges of G_d are retrieved using the query q, then integrated into a candidate bipartite subgraph G_c. (2) The most query-relevant nodes in G_c are identified as seed nodes. New nodes from G_{init} are expanded from the seed nodes, forming an expanded graph G_l. (3) LLM performs aggregation over restored tables to identify new relevant table rows, and then eliminates irrelevant passages.

4.2 Query-relevant Node Expansion

Query-relevant node expansion process aims to identify additional query-relevant edges, including the edges that have not been present within G_d. We perform the expansion process at the node level, which is the most fine-grained level. This is to address the issue that early fusion inevitably includes query-irrelevant nodes in the candidate subgraph. We formalize the process as finding a set of edges that meet the following objective function:

$$\underset{(u,v)\in E^* \wedge u \in V_c}{\arg\max} \quad p(u,v|q) = p(v|u,q)p(u|q) \qquad (5)$$

Here, u represents a node in the candidate graph G_c, and v represents a node adjacent to u in the complete bipartite graph G^*. The complete bipartite graph $G^* = (\mathcal{C}_{init}, E^*, \Phi, \Gamma, \Lambda_{init})$ contains all possible edges between table segments and passages. Naively solving this objective requires calculating the similarity score between the query and every possible edge in the complete graph, which is infeasible. To address this, we adopt a beam search approach and model it as a two-step process, as illustrated in Figure 3.

(1) Seed node selection: This step composes the first iteration of the beam search, corresponding to finding the set of nodes that show the highest $p(u|q)$. We calculate $p(u|q)$ for each $u \in V_c$ to identify the top-b (i.e., beam width) nodes that contain the most relevant information to the query. The probability $p(u|q)$ is determined by calculating the semantic similarity between the query and each node u in G_c, then normalizing the scores using a softmax function. This similarity is computed through a fine-tuned all-to-all interaction-based node reranker g_n.

(2) Seed node expansion: This step composes the second iteration of the beam search, aiming to find the set of edges (u, v) which show the highest $p(u, v|q)$. It is done by computing $p(v|u, q)$ for each node v connected to a seed node u in the complete bipartite graph G^*. This conditional probability is calculated using the expanded query retrieval technique (Xiong et al.). In this technique, the score function is expressed based on the expanded query as $sim([q; \Gamma(u)], v)$, and it is calculated by two late interaction models: $sim([q; \Gamma(u)], v; f_{P\to S})$ for a table-segment-typed expanding node and a passage-typed seed node, and $sim([q; \Gamma(u)], v; f_{S\to P})$ for the opposite. The calculated scores are normalized using a softmax function to compute $p(v|u, q)$. Finally, $p(u, v|q)$ is calculated for the (u, v) pairs following Equation 5, and the top-b edges with the highest scores are finally selected. The (u, v) pairs are added to E_c along with the nodes v to V_c, forming the expandede bipartite graph $G_l = (V_l, E_l, \Phi, \Gamma, \Lambda_l)$. The scores for each new edge is calculated using the reranker g_e as one used in § 4.1. Detailed explanation for fine-tuning g_n, $f_{P\to S}$ and $f_{S\to P}$ can be found in Appendix § A.2 and § A.3.

4.3 Star-based LLM Refinement

The main goal of this step is to retrieve query-relevant information which is challenging to find using semantic similarity alone, by leveraging the logical inference capabilities of LLMs. Selecting the optimal format or unit for presenting the graph G_l to the LLM is non-trivial. We explored two approaches: (i) providing the entire graph G_l in a single prompt to return the relevant set of nodes and (ii) decomposing G_l into star graphs, with each star graph generating its own set of relevant nodes. The latter approach proved to be 12.4% more effective, leading us to adopt star graphs as the unit for logical inference (§ 5.3). An overview of the process and few-shot prompts is in Appendix B and § H. The process consists of two phases: *column-wise aggregation* and *passage verification*.

(1) Column-wise Aggregation: This step aims to infer the correct result rows for table aggregation operations, as exemplified in Figure 1(c), making it possible that the corresponding rows exist in G_l.

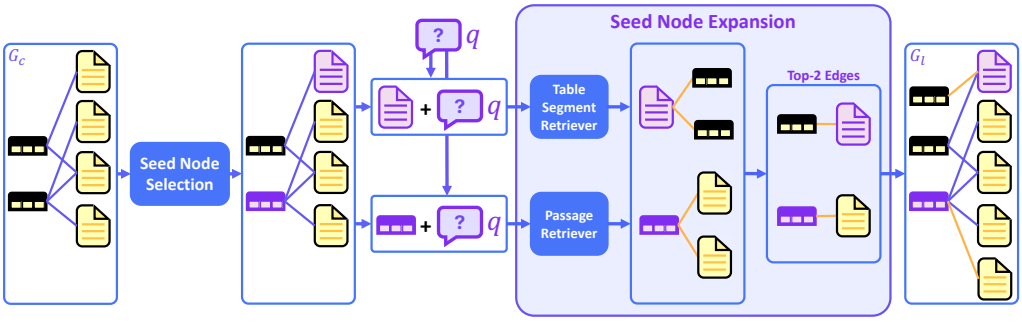


Figure 3: The overall procedure of query-relevant node expansion. The beam width b is set as 2 in this example. The purple-colored nodes indicate the selected seed nodes.

Since not every query requires aggregation, we first prompt the LLM to determine whether the input query necessitates an aggregation operation. If the query is classified as an aggregation query, it first restores the original from each table segment using mapping function $\mathcal{M}$. The restored tables are then provided to the LLM in the format of star graph along with the query. The LLM performs the aggregation and returns the rows corresponding to the aggregation result. The returned rows (i.e., table segments) are subsequently added to G_l along with their associated passages to generate G_a.

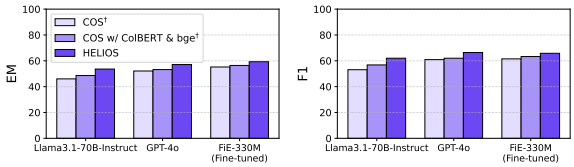
(2) Passage Verification: The edges of G_a may contain lots of hard negatives, as they comprise the edges retrieved from the edge retrieval, node expansion and column-wise aggregation step. The passage verification step aims to identify the passages relevant to answering the query. Similar to the column-wise aggregation step, we provide G_a to the LLM in the form of star graphs, units that contain multi-hop relationships. The LLM performs a binary verification to determine whether each edge is relevant to the query, without recalculating their scores. As a result, query-irrelevant edges are removed, yielding a refined edge-scored graph G_q.

We transform the graph G_q into a sequence of edges, as our reader requires a serialized token sequence as input. We use the mapping function Λ_c to sort each edge e by its similarity to the query $\Lambda_c(e)$, and the top k edges are returned.

5 Experiments

Hardware and Software Settings. We conducted our experiments on a machine with AMD EPYC 7313 CPU and 2TB of RAM with the OS of Rocky Linux release 8.7 and 4 A100-80GB GPUs.

Competitors. `HELIOS` is compared with the SOTA methods. The early fusion methods include `Fusion-Retriever` (Chen et al., 2020a), `OTTeR` (Huang et al., 2022), and `DoTTeR` (Kang et al., 2024). The late fusion approaches include `Iterative-Retriever` (Chen et al., 2020a), `CORE` (Ma et al., 2022), and `COS` (Ma et al., 2023).



(a) EM Score (b) F1 Score

Figure 4: End-to-end QA accuracy comparison across different readers for dev set of OTT-QA

Datasets. We conduct experiments on two datasets: `OTT-QA` (Chen et al., 2020a) and `MultimodalQA` (`MMQA`) (Talmor et al.). `OTT-QA` serves as the primary benchmark, as it is the only dataset designed for open-domain QA over tables and text, comprising 400K tables, 5M passages, and 42K training QA pairs, with 2K QA pairs each for development and testing. `MMQA` is a multi-hop QA dataset spanning images, passages, and tables. While not fully aligned with our task, we use it as a supplementary benchmark to assess generalizability. We exclude image-based questions and conduct experiments in an open setting using its full corpus (10K tables, 210K passages, and 1.3K QA pairs) without reference candidates. More detailed experimental settings are in Appendix § C.

5.1 Main Results

We evaluate retrieval accuracy using top-k Answer Recall (AR@k), nDCG@k, and Hits@4K, alongside end-to-end performance metrics: Exact Match (EM) and F1 score. AR@k measures the proportion of queries where the correct answer appears in the top-k retrieved edges (Ma et al., 2023), while nDCG@k measures the ranking quality considering both the relevance and the position of the retrieved edges. Hits@4K evaluates whether the answer span remains within the top 4096 tokens after linearizing ranked edges (Chen et al., 2020a). To an-

Model	AR@2	AR@5	AR@10	AR@20	AR@50	nDCG@50	HITS@4k
Iterative Retriever	–	–	–	–	–	–	27.2
Fusion Retriever	–	–	–	–	–	–	52.4
OTTeR†	31.4	49.7	62.0	71.8	82.0	25.9	70.1
DoTTeR†	31.5	51.0	61.5	71.9	80.8	26.7	70.3
CORE†	35.3	50.7	63.1	74.5	83.1	25.4	77.2
COS†	44.4	61.6	70.8	79.5	87.8	33.6	81.8
COS w/ ColBERT & bge†	49.6	68.2	78.7	85.0	91.7	36.5	85.9
HELIOS	**63.3**	**76.7**	**85.0**	**90.4**	**94.2**	**47.0**	**91.8**

Table 1: Retrieval accuracy on OTT-QA's dev set. Results marked with † indicate reproduced values.

Model	AR@2	AR@5	AR@10	AR@20	AR@50
COS†	50.7	59.7	67.1	72.4	79.5
HELIOS	**70.5**	**77.8**	**81.0**	**82.6**	**86.2**

Table 2: AR@k on MMQA's dev set.

Algorithm	Dev		Test	
	EM	F1	EM	F1
OTTeR	37.1	42.8	37.3	43.1
DoTTeR	37.8	43.9	35.9	42.0
CORE	49.0	55.7	47.3	54.1
COS	56.9	63.2	54.9	61.5
HELIOS	**59.3**	**65.8**	**57.0**	**64.3**

Table 3: End-to-end QA accuracy on OTT-QA.

Algorithm	AR@2	AR@5	AR@10	AR@20	AR@50	nDCG@50	EM	F1
HELIOS	**63.3**	76.7	85.0	**90.4**	94.2	47.0	59.3	65.8
w/ Finetuned SLR	63.2	**76.8**	**85.1**	90.3	**94.8**	**47.6**	**59.4**	**65.9**
w/o QNE	62.5	74.7	82.7	88.4	92.7	45.1	56.9	63.2
w/o SLR	60.0	75.2	84.7	90.1	94.6	46.5	59.0	65.7

Table 4: Retrieval accuracy of OTT-QA's dev set for HELIOS's design factors.

alyze the impact of retrieval accuracy on question-answering performance, we conduct end-to-end evaluations using EM and F1 scores. We select $k \in \{2, 5, 10, 20, 50\}$ based on evaluation protocols of state-of-the-art early and late fusion models (Kang et al., 2024; Ma et al., 2023). If $\mathcal{E}_q$ contains fewer edges than the retrieval target, we incorporate edges removed during the star-based LLM refinement stage to ensure a comprehensive assessment.

Table 1 shows the retrieval accuracy of HELIOS on OTT-QA's development set. HELIOS consistently outperforms all competitors on AR@k across different k values. It outperforms the state-of-the-art COS model by an average of 19.0% in AR, with the performance gap widening as k decreases. At $k = 2$, HELIOS achieves as much as 42.6% higher answer recall than COS. This improvement is further reflected in nDCG@50, where HELIOS exhibits a 39.9% gain. Additionally, the Hits@4K metric shows a 12.2% improvement over COS. We report an enhanced version of COS, denoted as COS w/ ColBERT & bge, which incorporates ColBERT retrievers and a bge reranker. Since HELIOS employs late-interaction retrieval, which generally outperforms single-embedding retrievers, we ensure COS uses the same retriever and reranker for a fair comparison. While this modification improves nDCG@50 by 8.6% over the original COS, HELIOS still outperforms the enhanced version of COS by a substantial margin of 28.8%.

Table 2 shows the retrieval accuracy of HELIOS and COS on MMQA's development set. HELIOS maintains its superior performance, achieving an average improvement of 20.9% in AR across all k values. We claim that this result indicates the robustness of HELIOS across different datasets.

Table 3 shows the end-to-end QA accuracy of HELIOS and COS on OTT-QA's development and test sets. Following COS, we used a Fusion-in-Encoder (FiE) model (Kedia et al., 2022) fine-tuned on the OTT-QA dataset. For a fair comparison, we provided the reader with 50 edges, matching the number of edges used in COS. The results indicate that compared to the COS model, our approach improved both EM and F1 scores by 4.2% and 4.1% on the development set, as well as by 3.8% and 4.6% on the test set, respectively.

Figure 4 presents the end-to-end QA accuracy of HELIOS, COS, and COS w/ ColBERT & bge across different reader models, including Llama-3.1-70B-Instruct (Dubey et al., 2024) and GPT-4o (Hurst et al., 2024). For each algorithm, we report the results of the value of k that yields the highest performance, where $k \in \{2, 5, 10, 20, 50\}$. Detailed results for other values of k are provided in the Appendix § C.3. HELIOS improved the performance of all readers, achieving an average EM score improvement of 7.5% and an average F1 score increase of 6.6% compared to COS w/ ColBERT & bge. Through these results, we claim that our well-retrieved documents are capable of enhancing the effectiveness of various readers. Examples of the prompts for the readers are in Appendix § H.

5.2 Ablation Study

We performed an ablation study to assess the contribution of query-relevant node expansion (QNE)

Retrieval Unit	AR@2	AR@5	AR@10	AR@20	AR@50	nDCG@50
Node	29.3	47.4	58.8	68.5	79.5	23.8
Star Graph	37.9	57.4	66.9	76.4	84.5	28.5
Edge	**49.1**	**63.1**	**70.6**	**77.6**	**85.1**	**34.2**

Table 5: Retrieval accuracy across different units

and star-based LLM refinement (SLR) to retrieval accuracy. In `w/o QNE`, we removed the QNE and `HELIOS` passes the candidate bipartite subgraph G_c directly to the SLR. In `w/o SLR`, the SLR was removed and `HELIOS` decomposes the expanded graph G_l into a list of edges.

As in Table 4, we found that removing the QNE module led to an average performance degradation of 2.1% in AR across all k values and 4.2% in nDCG@50. Additionally, excluding the QNE module resulted in a 4.2% decrease in EM and a 4.1% decrease in F1 scores. This highlights QNE's role in generating query-relevant edges missed by offline entity linking. For the `w/o SLR`, we observed a noticeable drop in AR@2, AR@5, AR@10, AR@20, nDCG@50, EM score, and F1 score, with accuracy decreases of 5.5%, 2.0%, 0.4%, 0.3%, 1.1%, 0.5%, and 0.2%, respectively. This suggests that SLR module helps accurately identify query-relevant nodes in complex queries requiring logical inference, particularly when k is small. Interestingly, for AR@50, `w/o SLR` slightly outperformed `HELIOS` by 0.4%, likely due to LLM hallucinations. Specifically, we observed 12 instances where the LLM failed to select the correct passage despite being provided with the ground truth passage. We present the qualitative analysis results in Appendix § D.

To mitigate hallucinations, we fine-tuned the SLR module using a training dataset synthesized with GPT-4o, incorporating aggregate query detection, column-wise aggregation, and passage verification tasks. As a result, the fine-tuned SLR module correctly selected the passage for 11 out of the 12 previously failed instances. In the remaining case, the LLM correctly identified the relevant document, but it was not included in the provided annotations. Details on this case are provided in Figure 8(c) of Appendix § D. The fine-tuning process is detailed in Appendix § A.4.

5.3 Impact of Granularity to Accuracy

We analyzed the impact of retrieval unit granularity on accuracy by comparing three versions of our subgraph retriever: (i) Node retriever: Retrieves table segments first, then links related passages via entity linking. (ii) Star graph retriever: Retrieves star graphs and integrates them into a graph. (iii) Edge retriever: Retrieves edges and integrates them to construct a graph. To ensure a fair comparison, we used the `ColBERTv2` baseline without fine-tuning.

As shown in Table 5, edge-based retrieval consistently outperformed the others. On average across all k values, it exceeded star graph- and node-based retrieval by 6.9% and 12.4%, respectively, and for nDCG@50, by 20% and 43.7%. This highlights edge-based retrieval's ability to provide richer information while minimizing information loss, striking an effective balance compared to the others.

We further evaluated two refinement strategies using an LLM: a full graph prompt versus individual prompts for each star graph. The latter, which reduced irrelevant information in prompts, improved nDCG@50 by 12.4% over the full graph setting (41.8), reducing hallucinations risks.

Algorithm	Execution Time (s)	nDCG@50
`CORE`	4.13	25.4
`COS`	3.75	33.6
`COS w/ ColBERT & bge`	5.46	36.5
`HELIOS`	5.14	47.0
`HELIOS w/ Finetuned SLR`	4.76	**47.6**
`w/o SLR`	**2.16**	46.5

Table 6: Algorithm execution time comparison

5.4 Algorithm Execution Time

As shown in Table 6, `HELIOS` finds a sweet spot between the increase in algorithm runtime (1.37×) and the increase in retrieval accuracy (39.9% nDCG@50). We further found out that fine-tuning LLM used in the SLR module can reduce the algorithm runtime 1.26× to that of `COS`, while boosting nDCG@50 to 41.7%. Interestingly, runtime of `HELIOS` can be reduced to at 0.57× of `COS` by removing the SLR module, yet it shows a 38.4% nDCG@50 gain. We reason this with `HELIOS` employing a beam search algorithm (beam width $b = 10$), whereas `COS` performs expanded query retrieval on all 200 retrieved table segments, corresponding to have a beam width of 20 times larger size.

6 Conclusion

We presented `HELIOS`, a novel table-text retrieval method that harmonizes the strengths of both early and late fusion techniques while incorporating LLM reasoning. It addresses the limitations of competitors by introducing a multi-granular retrieval system that optimally balances granularity across retrieval stages. Experiments on OTT-QA show that it surpasses SOTA models, achieving a 42.6% AR@2 improvement and a 39.9% nDCG@50 gain.

7 Limitations

Our approach currently focuses on the connections between table segments and passages. In future work, we aim to extend our method to a more general types of connections between nodes of arbitrary types, such as images and videos. While HELIOS achieves high retrieval accuracy, there is room to improve end-to-end accuracy performance.

References

BAAI. 2024. bge-reranker-v2-minicpm-layerwise.

Wenhu Chen, Ming-Wei Chang, Eva Schlinger, William Wang, and William W Cohen. 2020a. Open question answering over tables and text. *arXiv preprint arXiv:2010.10439*.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Wang. 2020b. Hybridqa: A dataset of multi-hop question answering over tabular and textual data. *arXiv preprint arXiv:2004.07347*.

Zhiyu Chen, Mohamed Trabelsi, Jeff Heflin, Dawei Yin, and Brian D Davison. 2021. Mgnets: multi-graph neural networks for table search. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 2945–2949.

Reinhard Diestel. 2024. *Graph theory*. Springer (print edition); Reinhard Diestel (eBooks).

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

Konrad Gadzicki, Razieh Khamsehashari, and Christoph Zetzsche. 2020. Early vs late fusion in multimodal convolutional neural networks. In *2020 IEEE 23rd international conference on information fusion (FUSION)*, pages 1–6. IEEE.

Junjie Huang, Wanjun Zhong, Qian Liu, Ming Gong, Daxin Jiang, and Nan Duan. 2022. Mixed-modality representation learning and pre-training for joint table-and-text retrieval in openqa. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4117–4129.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Deokhyung Kang, Baikjin Jung, Yunsu Kim, and Gary Geunbae Lee. 2024. Denoising table-text retrieval for open-domain question answering. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 4634–4640.

Akhil Kedia, Mohd Abbas Zaidi, and Haejun Lee. 2022. Fie: Building a global probability space by leveraging early fusion in encoder for open-domain question answering. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4246–4260.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Kaixin Ma, Hao Cheng, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. 2022. Open-domain question answering via chain of reasoning over heterogeneous knowledge. *arXiv preprint arXiv:2210.12338*.

Kaixin Ma, Hao Cheng, Yu Zhang, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. 2023. Chain-of-skills: A configurable model for open-domain question answering. *arXiv preprint arXiv:2305.03130*.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2021. Colbertv2: Effective and efficient retrieval via lightweight late interaction. *arXiv preprint arXiv:2112.01488*.

Cees GM Snoek, Marcel Worring, and Arnold WM Smeulders. 2005. Early versus late fusion in semantic video analysis. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 399–402.

Alon Talmor, Ori Yoran, Amnon Catav, Dan Lahav, Yizhong Wang, Akari Asai, Gabriel Ilharco, Hannaneh Hajishirzi, and Jonathan Berant. Multimodalqa: complex question answering over text, tables and images. In *International Conference on Learning Representations*.

Fei Wang, Kexuan Sun, Muhao Chen, Jay Pujara, and Pedro Szekely. 2021. Retrieving complex tables with multi-granular graph representation learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1472–1482.

Yifan Wang, Xing Xu, Wei Yu, Ruicong Xu, Zuo Cao, and Heng Tao Shen. 2022. Comprehensive framework of early and late fusion for image–sentence retrieval. *IEEE MultiMedia*, 29(3):38–47.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, et al. Chain-of-table: Evolving tables in the reasoning chain for table understanding. In *The Twelfth International Conference on Learning Representations*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302.

Wenhan Xiong, Xiang Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Scott Yih, Sebastian Riedel, Douwe Kiela, et al. Answering complex open-domain questions with multi-hop dense retrieval. In *International Conference on Learning Representations*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Ruize Yuan, Xiang Ao, Li Zeng, and Qing He. 2024. Drama: Dynamic multi-granularity graph estimate retrieval over tabular and textual question answering. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 5365–5375.

Qin Zhang, Shangsi Chen, Dongkuan Xu, Qingqing Cao, Xiaojun Chen, Trevor Cohn, and Meng Fang. 2023. A survey for efficient open domain question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14447–14465, Toronto, Canada. Association for Computational Linguistics.

Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2024. Tablellama: Towards open large generalist models for tables. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6024–6044.

A Training Scheme

A.1 Edge Retriever and Reranker

The training scheme for our encoder f_e follows the methodology outlined in `ColBERTv2` (Santhanam et al. 2021), leveraging a combination of in-batch negative loss and knowledge distillation loss to train the model. Specifically, the in-batch negative loss treats the edges corresponding to other queries within the same batch as negative samples. This approach calculates a contrastive loss between the positive and negative edges. In constructing the training dataset, it is crucial to have both positive and negative edges for each query. To define the positive edge, we use passages containing the answer and the associated table segments as ground truth and denoted as x_{gt}. Conversely, negative edges are constructed by combining hard negative tables and passages from prior work (Ma et al. 2023) with in-batch negative edges and are denoted as $n(q)$. The contrastive loss L_{cl} is represented as follows:

$$L_{cl} = -\sum_{(q,x_{gt})} \log \frac{exp(s(q,x_{gt}))}{exp(s(q,x_{gt})) + \sum_{z\in n(q)} exp(s(q,z))} \tag{6}$$

The knowledge distillation process refines the edge encoder using a teacher-student model setup. The distillation loss is computed based on the KL divergence between the score distribution generated by the teacher model and the training encoder. The training was conducted for 1 epoch with a batch size of 512 and a learning rate of 1e-5. We employed a cosine learning rate scheduler with a 40 warmup steps.

Here, the teacher model is the all-to-all interaction reranker g_e fine-tuned with the contrastive loss, which serves as a more precise reference for edge relevance. This method ensures that the encoder learns from a more sophisticated model, improving its capacity to accurately rank edges based on the query. The training was conducted for 1 epoch with a batch size of 96 and a learning rate of 2e-4. We employed a cosine learning rate scheduler with a warmup ratio of 0.1.

A.2 Node Reranker

The training method for the node reranker g_n is identical to that of the edge reranker g_e. For constructing the training dataset, we utilize the `OTT-QA` dataset (Chen et al., 2020a). Positive nodes are defined as those directly connected to the nodes

that contain the correct answer in OTT-QA. In contrast, negative nodes are selected from the set of nodes retrieved through edge-based bipartite subgraph retrieval, excluding any nodes connected to the answer-containing nodes. The training was conducted for 1 epoch with a batch size of 96 and a learning rate of 2e-4. We employed a cosine learning rate scheduler with a warmup ratio of 0.1.

A.3 Expanded Query Retrievers

The training scheme for our expanded query retrievers $f_{S\to P}$, $f_{P\to S}$ also follows the methodology outlined in ColBERTv2 (Santhanam et al. 2021). To construct the training dataset, we generated triples consisting of the expanded query, positive node, and negative node. Expanded queries were created by incorporating nodes that are connected to the node containing the answer. Positive nodes consist of the nodes that contain the answer. Negative nodes are constructed using hard negative nodes as outlined in prior work (Ma et al. 2023). The training was conducted for 1 epoch with a batch size of 512 and a learning rate of 1e-5. We employed a cosine learning rate scheduler with a 40 warmup steps.

A.4 Star-based LLM Refinement

We fine-tuned the Llama-3.1-Instruct 8B model (Dubey et al., 2024) using an instruction dataset synthesized with GPT-4o (Hurst et al., 2024). The dataset includes tasks for aggregate query detection, column-wise aggregation, and passage verification, and training was conducted for 1 epoch with a batch size of 128 and a learning rate of 2e-5. We employed a linear learning rate scheduler with a warmup ratio of 0.03.

To construct the training data, we applied edge-based bipartite graph retrieval and query-relevant node expansion on the OTT-QA training set, providing the results to GPT-4o for output generation. The final dataset was formed by combining these outputs with zero-shot prompts. Due to API cost constraints, we sampled 25% of the OTT-QA training dataset, resulting in 1,655 samples each for query detection and column-wise aggregation and 7,010 samples for passage verification.

B Star-based LLM Refinement Supplementary

In the Star-based LLM Refinement stage, the expanded graph from the Query-relevant Node Expansion (QNE) step undergoes further enhancement through the reasoning capabilities of LLMs. This stage consists of two primary operations: (1) aggregation of query-relevant table segments and (2) verification of query-irrelevant passages. The detailed process is illustrated in Figure 5.

C Experiment Supplementaries

C.1 Implementation Details

In our edge generation step (§ 4.1), we used the same named entity recognition and entity linking models used by COS (Ma et al., 2023). For the late-interaction edge retriever f_e (§ 4.1) and the expanded query retrievers $f_{P\to S}$ and $f_{S\to P}$ (§ 4.2), we employed ColBERTv2 (Santhanam et al., 2021) as the baseline model. For the all-to-all interaction edge reranker g_e (§ 4.1) and node reranker g_n (§ 4.2), we used the bge-reranker-v2-minicpm-layerwise (BAAI, 2024), specifically utilizing layer 24 as the baseline model. Lastly, for star-based LLM refinement (§ 4.3), we used Llama-3.1-8B-Instruct (Dubey et al., 2024) as the large language model. In our experiments, The value of k_1 for the edge retriever f_e was set to 400. Since COS selects the top-200 nodes as seed nodes, we fixed k_2 for the edge reranker g_e to 100 to ensure a fair comparison. We evaluated HELIOS on the OTT-QA's development set (2,214 examples) using four A100-80GB GPUs, which required approximately 3.4 hours.

C.2 Parameter Sensitivity of Retriever

We explored the impact of varying beam width b on retrieval accuracy in terms of AR@50. The beam width directly influences the number of expanded nodes (§ 4.2). We experimented with beam widths of 0, 2, 5, 10, 25, 50 and measured the corresponding changes in AR@50.

Figure 6 illustrates the change in AR@50. We observed that AR@50 was improved by 1.7% as the beam width monotone increased from 0 to 10. This indicates that larger beam widths lead to more accurate node augmentations by performing a more exhaustive search across the expanding node space. Interestingly, when the beam size increased to 50, AR@50 decreased slightly by 0.4% compared to beam size 10. This drop may be due to LLM hallucinations in the star-based LLM refinement (SLR) module, where irrelevant edges were added to G_l, causing the SLR to fail in selecting the correct query-relevant nodes. This observation highlights the importance of selectively expanding

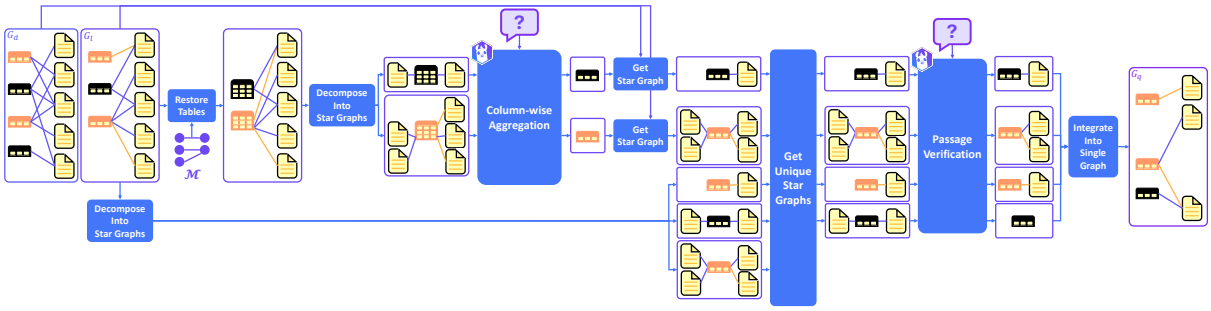


Figure 5: The overall process of star-based LLM refinement for queries classified as aggregation queries. Table segment nodes of the same color (black, orange) indicate segments that belong to the same original table.

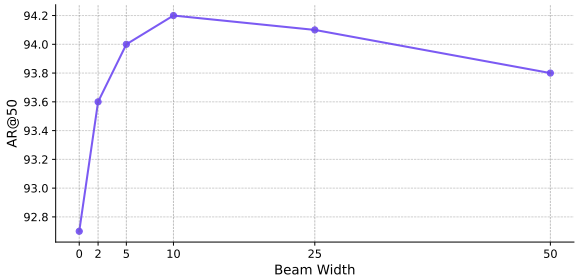
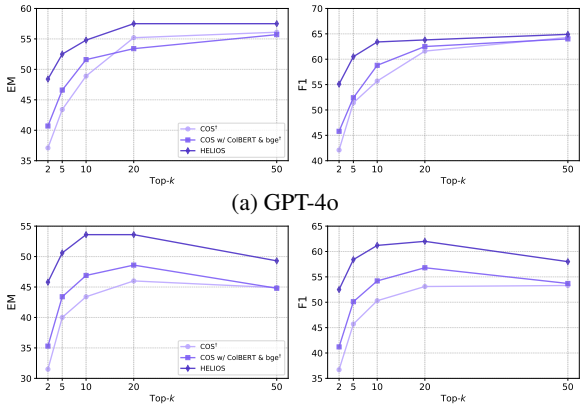


Figure 6: Change in AR@50 with varying beam width



(a) GPT-4o

(b) Llama3.1-Instruct-70B

Figure 7: End-to-end QA accuracy comparison across different top-k values on the OTT-QA dev set.

only the most probable nodes within the query-relevant node expansion module.

C.3 Parameter Sensitivity of Reader

Figure 7 presents the end-to-end QA accuracy of HELIOS, COS, and COS w/ ColBERT & bge across different reader models—Llama-3.1-70B-Instruct (Dubey et al., 2024) and GPT-4o (Hurst et al., 2024)—with varying top-k values ($k \in 2, 5, 10, 20, 50$). Due to budget constraints, we sampled 10% (221 out of 2,214) of the development set's QA pairs when evaluating accuracy variations for GPT-4o. As a result, HELIOS consistently achieves the highest AR@k across all k values. It improves the average EM score by 15.5% for Llama-3.1-70B-Instruct and 9.2% for GPT-4o compared to COS w/ ColBERT & bge.

D Qualitative Analysis

In this section, we present a qualitative analysis of HELIOS's Column-wise Aggregation module and Passage Verification module, with the results illustrated in Figure 8. The subfigures in Figure 8 showcase the performance and distinctive scenarios for each module: (a) highlights successful cases of the Column-wise Aggregation module, while (b), (c), and (d) demonstrate representative cases related to the Passage Verification module. For each subfigure, the query is depicted in dark blue, the data provided to submodule is shown in light blue, and the inference result from the LLM are encapsulated in a purple speech bubble with a llama icon.

Figure 8(a) shows a successful case of the column-wise aggregation module in resolving a complex query: identifying the birth date of the "most recent Segunda Liga Player of the Month." The essential part of answering this question was to recognize that the most recent player, Basilio Almeida, was honored in November 2009, as indicated in the SJPF Segunda Liga Player of the Month table. However, the initial data lacked the table segment containing the relevant row. The column-wise aggregation module reconstructed the table as shown in Figure 8(a) to include this missing information, enabling the system to restore the row with the necessary details. The LLM correctly inferred from the reconstructed table that the row corresponding to the most recent player was Row 4, based on the Year and Month columns. This lead HELIOS to accurately generate the final answer in this question, which is "12 August 1971."

Figure 8(b) shows a successful case of the pas-

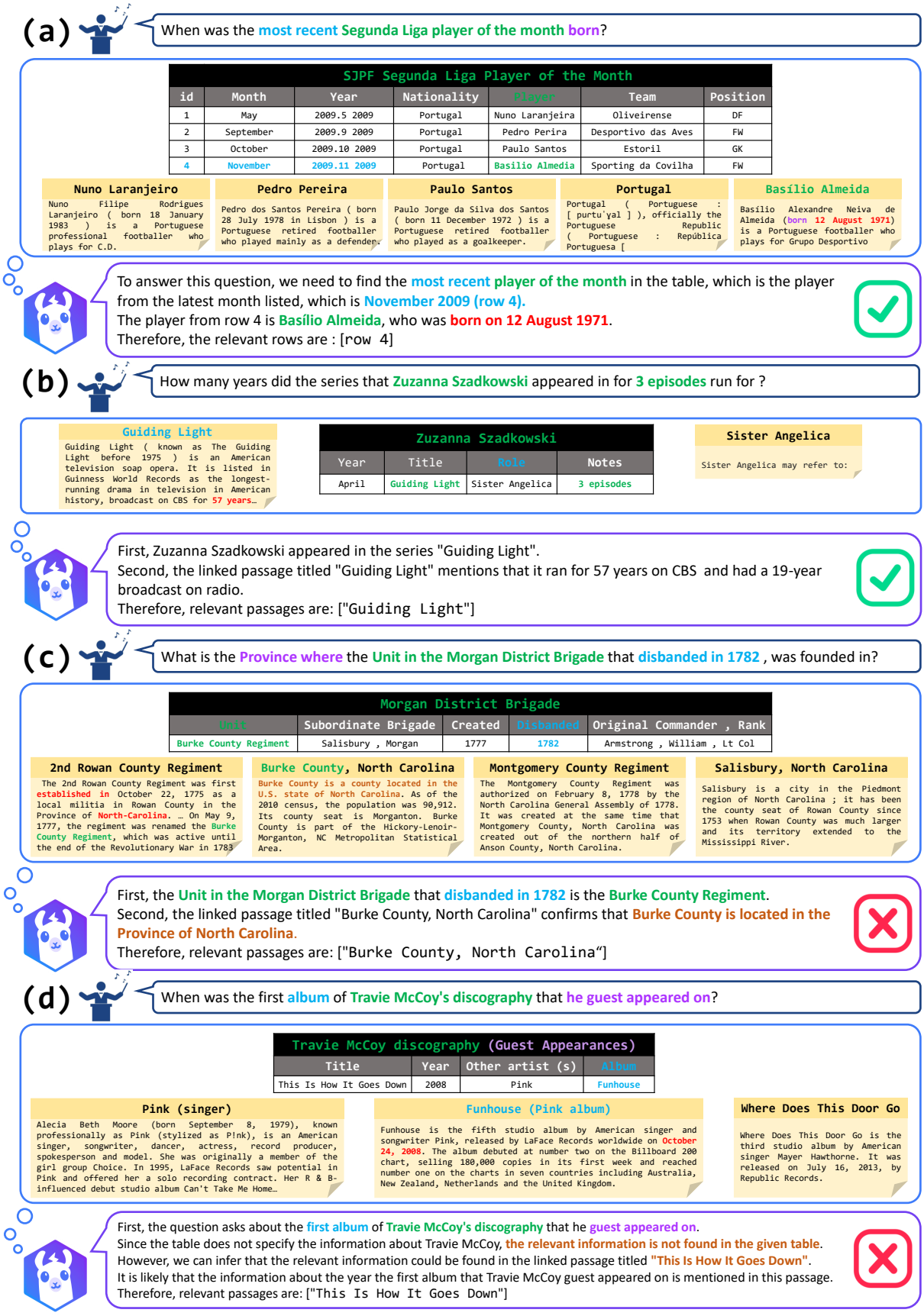


Figure 8: Qualitative analysis on four question-answer pairs. (a) A case where passage verification is successful. (b) A first case where passage verification has failed. (c) A second case where passage verification has failed. (d) A case where table aggregation is successful.

sage verification module in addressing the query, "How many years did the series that Zuzanna Szadkowski appeared in for 3 episodes run for?". The module was provided with a Zuzanna Szadkowski table summarizing her appearances and a set of associated passages. The "Notes" column of the table segment confirmed that she appeared in three episodes of the series Guiding Light. The module correctly identified the one mentioning Guiding Light among the provided passages, the one which indicated that the series was broadcast on CBS for 57 years. the module correctly verified that the passage using the passage's information noting its broadcast duration, leading to an accurate answer.

Figure 8(c) shows a failure case of the Passage Verification module when answering the query, "What is the province where the unit in the Morgan District Brigade that disbanded in 1782 was founded?". The module correctly identified 'Burke County Regiment' as relevant to the query by recognizing from the 'Morgan District Brigade' table segment that the 'Disbanded' column value was 1782. However, information related to this query was present in two passages: '2nd Rowan County Regiment' and 'Burke County, North Carolina'. The LLM incorrectly verified only 'Burke County, North Carolina' as relevant, likely due to its more plausible-sounding title, while overlooking the correct answer 'North-Carolina' in the passage titled '2nd Rowan County Regiment'. Consequently, the system produced an incorrect response, 'North Carolina'. This error highlights two problems: (i) a limitation of the LLM reasoning capability and (ii) an example case of the `OTT-QA` benchmark's wrong answer annotation.

Figure 6(d) shows another failure case of the passage verification module, this time for the query, "When was the first album of Travie McCoy's discography that he guest appeared on?". Prior retrieval results correctly introduced the ground truth table titled 'Travie McCoy discography (Guest Appearances)' to the passage verification module. However, the LLM incorrectly inferred that "the table does not specify the information about Travie McCoy" as seen in the second line of its response bubble. It then relied on its parameterized knowledge to wrongly verify a passage titled 'This Is How It Goes Down as relevant'. The correct answer 'Funhouse (Pink album)' was excluded from the final retrieved document set due to the verification error.

E Software and Data Licenses

The licenses for the software and datasets used in this paper are as follows:

- ColBERTv2: MIT License
- bge-reranker-v2-minicpm-layerwise: Apache 2.0 License
- LLaMA 3.1-8B-Instruct: LLaMA 3.1
- LLaMA 3.1-70B-Instruct: LLaMA 3.1
- Chain-of-Table: Apache 2.0 License
- TableLlama: MIT License
- COS: GPL-3.0 License
- OTT-QA: MIT License

All software and datasets were used strictly for research purposes and were not utilized in any non-research contexts, particularly for commercial applications. The datasets used in this study, OTT-QA and MultimodalQA, consist of English-language data sourced from the Wikipedia domain.

F AI Assistants

We used ChatGPT-4o (Hurst et al., 2024) to debug code efficiently, quickly identifying and resolving errors in our implementations. Additionally, we used it for rephrasing sentences in my writing to improve clarity and readability.

G Reproducibility Statement

OTTeR (Huang et al., 2022) and DoTTeR (Kang et al., 2024) were reproduced using the official code available at `OTTeR` and `DoTTeR`, respectively. COS (Ma et al., 2023) and CORE (Ma et al., 2022) were reproduced using the official code from `UDT-QA`. The source code, data, and other artifacts for `HELIOS` have been made available at `anonymous.4open.science`.

H Prompt Templates

For Star-based LLM refinement, we extended the prompt from `Chain-of-Table` (Wang et al.), originally designed for selecting relevant rows from tables, to support column-wise aggregation and passage verification. This extension enables the joint consideration of table segments and linked passages. For the LLM reader, we constructed the prompt based on the HybridQA prompt from `TableLlama` (Zhang et al., 2024).

Aggregation Query Classification

Using `f_agg()` API, return True to detect when a natural language query involves performing aggregation operations (max, min, avg, group by). Strictly follow the format of the below examples. Please provide your explanation first, then answer the question in a short phrase starting by 'Therefore, the answer is:'

Question: when was the third highest paid Rangers F.C. player born?
Explanation: The question involves finding the birth date of the third highest paid player, which requires aggregation to find the third highest paid player. Therefore, the answer is: `f_agg([True])`

Question: what is the full name of the Jesus College alumni who graduated in 1960?
Explanation: The question involves finding the full name of the alumni who graduated in 1960, which does not require aggregation. Therefore, the answer is: `f_agg([False])`

Question: how tall, in feet, is the Basketball personality that was chosen as MVP most recently?
Explanation: The question involves finding the most recent MVP winner, which requires aggregation to identify the relevant player. Therefore, the answer is: `f_agg([True])`

Question: what is the highest best score series 7 of Ballando con le Stelle for the best dancer born 3 July 1969?
Explanation: The question involves finding the highest score in a series for a specific dancer, which requires aggregation. Therefore, the answer is: `f_agg([True])`

Question: which conquerors established the historical site in England that attracted 2,389,548 2009 tourists?
Explanation: The question involves identifying the conquerors who established a historical site, which does not require aggregation. Therefore, the answer is: `f_agg([False])`

Question: what is the NYPD Blue character of the actor who was born on January 29, 1962?
Explanation: The question involves finding the character played by an actor born on a specific date, which does not require aggregation. Therefore, the answer is: `f_agg([False])`

Question: '`{question}`'
Explanation:

Column-wise Aggregation

Using `f_row()` API to select relevant rows in the given table and linked passages that support or oppose the question. Strictly follow the format of the below example. Please provide your explanation first, then select relevant rows in a short phrase starting by: *"Therefore, the relevant rows are:"*

```
/* table caption : list of rangers f.c. records and statistics
col : # | player | to | fee | date
row 1 : 1 | alan hutton | tottenham hotspur | 9,000,000 | 30 january 2008
row 2 : 2 | giovanni van bronckhorst | arsenal | 8,500,000 | 20 june 2001
row 3 : 3 | jean-alain boumsong | newcastle united | 8,000,000 | 1 january 2005
row 4 : 4 | carlos cuellar | aston villa | 7,800,000 | 12 august 2008
row 5 : 5 | barry ferguson | blackburn rovers | 7,500,000 | 29 august 2003 */
/* Passages linked to row 1:
- Alan Hutton: Alan Hutton (born 30 November 1984) is a Scottish former
professional footballer, who played as a right back. Hutton started his career
with Rangers, and won the league title in 2005.
- Tottenham Hotspur F.C.: Tottenham Hotspur Football Club, commonly referred to
as Tottenham or Spurs, is an English professional football club in Tottenham,
London, that competes in the Premier League. */
/* Passages linked to row 2:
- Giovanni van Bronckhorst: Giovanni Christiaan van Bronckhorst (born 5 February
1975), also known by his nickname Gio, is a retired Dutch footballer and
currently the manager of Guangzhou RF. */
/* Passages linked to row 3:
- Jean-Alain Boumsong: Jean-Alain Boumsong Somkong (born 14 December 1979) is
a former professional football defender, including French international.
- Newcastle United F.C.: Newcastle United Football Club is an English
professional football club based in Newcastle upon Tyne, Tyne and Wear, that
plays in the Premier League, the top tier of English football. */
```

Question: 'When was the third highest paid Rangers F.C . player born ?'
Explanation: The third-highest paid Rangers F.C. player, Jean-Alain Boumsong (row 3). *Therefore, the relevant rows are:* `f_row([row 3])`'

```
/* '{table}' */

/* '{linked_passages}' */
```

Question: '{question}'
Explanation:

Passage Verification

Using `f_passage()` API to return a list of passage titles that are relevant to the question, even if they are only partially related. Strictly follow the format of the below example. Please provide your explanation first, then return a list of passages in a short phrase starting by: *"Therefore, relevant passages are:"*

```
/* table caption : List of politicians, lawyers, and civil servants educated at Jesus College, Oxford
col : Name | M | G | Degree | Notes
row 1 : Lalith Athulathmudali | 1955 | 1960 | BA Jurisprudence (2nd, 1958), BCL (2nd, 1960) | President of the Oxford Union (1958); a Sri Lankan politician; killed by the Tamil Tigers in 1993 */
/* List of linked passages: ["Law degree", "Oxford Union", "Lalith Athulathmudali"]
Title: Lalith Athulathmudali. Content: Lalith William Samarasekera Athulathmudali, PC (Sinhala; 26 November 1936 - 23 April 1993), known as Lalith Athulathmudali, was a Sri Lankan statesman. He was a prominent member of the United National Party, who served as Minister of Trade and Shipping; Minister of National Security and Deputy Minister of Defence; Minister of Agriculture, Food and Cooperatives, and finally Minister of Education.
Title: Law degree. Content: A law degree is an academic degree conferred for studies in law. Such degrees are generally preparation for legal careers; but while their curricula may be reviewed by legal authority, they do not themselves confer a license. A legal license is granted (typically by examination) and exercised locally; while the law degree can have local, international, and world-wide aspects.
Title: Oxford Union. Content: The Oxford Union Society, commonly referred to simply as the Oxford Union, is a debating society in the city of Oxford, England, whose membership is drawn primarily from the University of Oxford. Founded in 1823, it is one of Britain's oldest university unions and one of the world's most prestigious private students' societies. The Oxford Union exists independently from the university and is separate from the Oxford University Student Union. */
```

Question: What is the full name of the Jesus College alumni who graduated in 1960?
Explanation: First, Lalith Athulathmudali graduated in 1960. Second, the linked passage titled "Lalith Athulathmudali" confirms his full name. *Therefore, relevant passages are:* `f_passage(["Lalith Athulathmudali"])`

```
/* '{table}' */

/* '{linked_passages}' */
```

Question: '`{question}`'
Explanation:

Hybrid Question Answering

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

Instruction:
This is a hybrid question answering task.
The goal of this task is to answer the question given tables and passages.
Strictly follow the format of the below examples. Please provide your explanation first, then answer the question in a short phrase starting by: *'Therefore, the answer is:'*

Examples:
`Title : List of politicians, lawyers, and civil servants educated at Jesus College, Oxford`
`col : Name | M | G | Degree | Notes`
`row 1 : Lalith Athulathmudali | 1955 | 1960 | BA Jurisprudence ( 2nd , 1958 ) , BCL ( 2nd , 1960 ) | President of the Oxford Union ( 1958 ) ; a Sri Lankan politician ; killed by the Tamil Tigers in 1993`
`row 2 : Neal Blewett ( HF ) | 1957 | 1959 | BA PPE ( 2nd ) | Member of the Australian House of Representatives ( 1977-1994 ) , Government Minister ( 1983-1994 ) , High Commissioner to the UK ( 1994-1998 )`
`row 3 : Joseph Clearihue | 1911 | 1914 | BA Jurisprudence (2nd , 1913) , BCL (3rd , 1914) | Canadian Rhodes scholar ; later became a member of the Legislative Assembly of British Columbia and a county court judge ; also chairman of the council of Victoria College , British Columbia (which became the University of Victoria under his leadership)`

`Passages linked to row 1:`
`- [Lalith Athulathmudali](https://en.wikipedia.org/wiki/Lalith_Athulathmudali) Lalith William Samarasekera Athulathmudali , PC (26 November 1936 - 23 April 1993) , known as Lalith Athulathmudali , was Sri Lankan statesman . He was a prominent member of the United National Party , who served as Minister of Trade and Shipping ; Minister National Security and Deputy Minister of Defence.`
`Passages linked to row 3:`
`- [Joseph Clearihue](https://en.wikipedia.org/wiki/Joseph_Clearihue) Joseph Badenoch Clearihue ( 1887-1976 ) was a Canadian lawyer , judge , academic and politician .`

Question: What is the full name of the Jesus College alumni who graduated in 1960?
Explanation: Lalith Athulathmudali graduated in 1960, and his full name is Lalith William Samarasekera Athulathmudali. Therefore, the answer is: Lalith William Samarasekera Athulathmudali.

Input:
Here are the table and passages to answer this question. Please provide your explanation first, then answer the question in a short phrase starting by *'Therefore, the answer is:'*

`/* '{table_and_linked_passages}' */`

Question: '`{question}`'
Explanation: