HyReT: A Unified Framework for Customizing Hybrid Retrieval

Anonymous ACL submission

Abstract

Hybrid retrieval systems, combining the strengths of both dense and sparse retrieval methods, are becoming increasingly popular in information retrieval. While customization of dense retrievers has shown promise, customizing the entire hybrid system remains an open challenge. This paper introduces **HyReT**, a novel framework for fine-tuning all key aspects of a hybrid retrieval system: the dense retriever, the sparse retriever, and the score aggregation function. By leveraging a set of relevant query-corpus pairs, **HyReT** optimizes the entire pipeline, leading to substantial improvements in retrieval accuracy. Our experiments on diverse English and multilingual datasets demonstrate the effectiveness of this unified approach. Specifically, with popular BEIR datasets, **HyReT** achieves a 15.9% performance gain over the best zero-shot hybrid methods and a 7.6% improvement compared to partially customized solutions.

1 Introduction

Large-scale retrieval plays a crucial role in various information seeking tasks, including open-domain question answering (Chen and Yih, 2020), web search (Shen et al., 2023), conversational AI (Zhao et al., 2020), and code generation (Jimenez et al., 2024). This involves efficiently identifying the most relevant documents or passages from a massive corpus of millions or even billions of entries in response to a given text query. Retrieval systems typically employ either dense or sparse approaches. Dense retrievers (Lee et al., 2024; Chen et al., 2024; Moreira et al., 2024), often based on Large Language Models (LLMs), leverage deep neural networks to capture semantic meaning. In contrast, sparse retrievers rely on keyword matching techniques like BM25 (Robertson et al., 2009) and TF-IDF (Aizawa, 2003), representing text with sparse vectors.

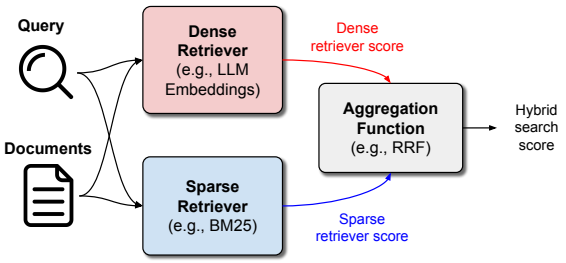

Figure 1: Hybrid retrieval systems utilize both dense and sparse retrievers to improve retrieval accuracy. The relevance scores from these two types of retrievers are then combined using an aggregation function to determine the final hybrid retrieval relevance scores.

While the increasing power of LLMs has fueled the popularity of dense retrievers for capturing complex semantic meaning, both dense and sparse retrieval methods offer unique advantages. Sparse retrievers excel at pinpointing crucial keywords, effectively localizing relevant information, which proves particularly valuable for reasoning tasks as demonstrated by recent benchmarks like BRIGHT (Su et al., 2024). Recognizing their complementary strengths, state-of-the-art retrieval systems, especially in large-scale real-world applications, often adopt a hybrid approach. These hybrid systems integrate signals from both dense and sparse retrievers, with effective aggregation methods playing a key role in maximizing overall performance (Bhagdev et al., 2008).

Customization, tailoring retrieval systems to specific datasets, has emerged as another critical trend in information retrieval. Customized retrievers often outperform generic, zero-shot approaches by achieving significant improvements in accuracy, latency, and cost. Recent research has demonstrated the effectiveness of customizing dense retrievers, with techniques like Search Adaptor (Yoon et al., 2023) and Matryoshka Adaptor (Yoon et al., 2024) yielding impressive gains. However, customizing the entire hybrid retrieval system, encompassing

Query (Orange highlights tokens that appear infrequently.)	Dense retriever	Sparse retriever	**HyReT**
(i) FoxO3a activation in neuronal death is mediated by reactive oxygen species (ROS). (ii) Neutrophil extracellular traps (NETs) are released by ANCA-stimulated neutrophils.	X	O	O
(iii) Obesity decreases life quality. (iv) The one-child policy has been successful in lowering population growth.	O	X	O

Table 1: Example cases where dense or sparse retrievers fail to identify relevant information, yet **HyReT** successfully does so (using SciFact data). Additional examples are provided in Appendix A.4.

dense and sparse retrievers along with their aggregation function, remains largely unexplored. This is a challenging endeavor due to factors such as limited access to dense retriever parameters, the complexities of handling high-dimensional sparse vectors, selecting suitable training objectives, and optimizing a model with heterogeneous inputs.

This paper introduces **HyReT**, a novel end-to-end framework for customizing hybrid retrieval systems. Going beyond existing work focused solely on dense retrievers, **HyReT** enables the optimization of all key components: dense retrievers, sparse retrievers, and the aggregation function. We present effective techniques for customizing each component individually and integrating them for optimal end-to-end performance. Especially, we propose a novel approach to customizing sparse retrievers, addressing the challenges of high-dimensional input that previously hindered customization. Through extensive experiments on diverse datasets, including multilingual collections, we demonstrate **HyReT**'s superiority over both zero-shot hybrid retrieval and approaches that customize only a subset of components. In some cases, **HyReT** achieves significant improvements with minimal training data, requiring only a few hundred query-corpus pairs (see Appendix. B.1). Furthermore, its adapter-based design allows for application to any combination of dense and sparse retrievers, regardless of access to model weights, establishing it as a versatile and widely applicable framework for hybrid retrieval customization.

2 Related Works

Dense embedding retrieval. Dense retrieval relies on high-quality embeddings to represent queries and documents as vectors for effective comparison. While specialized models like DPR (Karpukhin et al., 2020), ColBERT (Khattab and Zaharia, 2020) and Condenser (Gao and Callan, 2021) were once prevalent, LLMs based models now dominate dense retrieval with their superior embedding capabilities. Both industry (e.g., Google's gecko (Lee et al., 2024), OpenAI's embeddings (Neelakantan et al., 2022)) and academia (e.g., GTE (Li et al., 2023) and E5 (Wang et al., 2022)) are actively pushing the boundaries of embedding methods.

Customizing these dense retrievers for specific tasks has become another trend of research. While fine-tuning LLM encoders directly is possible (e.g., LoRA (Hu et al., 2021) or IA3 (Liu et al., 2022)), it often requires full model access and substantial training data. Alternatively, methods like Search-Adaptor (Yoon et al., 2023) offer a more flexible and data-efficient solution by introducing a trainable adaptor on top of extracted embeddings. Due to its advantages in model compatibility and minimal data requirements, we adopt Search-Adaptor as our primary customization method in this work.

Sparse embedding retrieval. Sparse retrievers rely on sparse embeddings, essentially vectors reflecting keyword frequencies, to identify relevant information. A high degree of keyword overlap between a query and a document indicates strong similarity in their sparse representations. BM25 (Robertson et al., 2009), a widely used sparse retrieval method, leverages inverse document frequency (IDF) to weigh keyword matches and calculate similarity scores. Building on this, SPLADE (Formal et al., 2021b,a) incorporates further regularization for enhanced performance.

However, customizing sparse retrievers for specific tasks presents challenges due to the high dimensionality of sparse vectors and their susceptibility to overfitting. Directly applying existing customization techniques designed for dense vectors, such as Search-Adaptor (Yoon et al., 2023), proves inefficient and exacerbates overfitting. To address this, we introduce a novel, model-agnostic adaptation method that effectively customizes sparse embeddings like BM25 and SPLADE using labeled data while mitigating overfitting.

Hybrid retrieval. Dense and sparse retriev-

ers each offer unique strengths, motivating the widespread adoption of hybrid retrieval systems in industry. These systems combine signals from both retriever types to maximize effectiveness (Bhagdev et al., 2008). While methods like Reciprocal Rank Fusion (RRF) (Cormack et al., 2009) offer a simple and effective way to aggregate scores, customizing the entire hybrid system remains largely unexplored. Existing approaches often rely on rigid techniques with limited flexibility, such as weighted sums of scores (e.g., (Gao et al., 2021) and DPR (Karpukhin et al., 2020)). In contrast, we propose a novel method that enables simultaneous customization of all key components in a hybrid retrieval system: the dense retriever, the sparse retriever, and the aggregation function. This approach offers a more comprehensive and adaptable solution for tailoring hybrid retrieval to specific tasks and datasets.

3 Hybrid Retrieval Tuning (HyReT)

Hybrid retrieval comprises three essential components: retrieval with dense embeddings, retrieval with sparse embeddings, and the aggregation function. In the following section, we provide an overview of each component and elucidate how they can be tuned using some labeled data. Subsequently, we introduce a unified training framework that integrates the customization of all three components.

3.1 Problem Formulation

Consider a dataset $\mathcal{D}$ containing queries $\mathcal{Q} = \{q_1, ..., q_N\} \in Q$, corpus documents $\mathcal{C} = \{c_1, ..., c_M\} \in C$, and their corresponding relevance scores. We denote the relevance score between query q_i and corpus document c_j as y_{ij}, where a positive value indicates relevance. The relevance scores can be either binary (relevant/not relevant) or continuous (degree of relevance). Our objective is to develop a retrieval system with a scoring function $f : Q \times C \to \mathbb{R}$ that effectively predicts the relevance between any given query and corpus document.

3.2 Dense retriever tuning

Dense retrievers, as dual encoder models (E^d), transform both queries (q_i) and corpus text (c_j) into numerical vectors ($qe_i^d = E^d(q_i), ce_j^d = E^d(c_j)$). These vectors capture the semantic meaning of the text, and their similarity (often measured by cosine similarity metric) indicates the relevance of a query to a document (i.e., $s_{ij}^d = \text{Sim}(qe_i^d, ce_j^d)$).

To further improve this process, we utilize Search-Adaptor (Yoon et al., 2023), a technique that can significantly improve retrieval performance utilizing labeled training data, without requiring access to the underlying model parameters. This flexibility enables Search-Adaptor to be applied on a wide range of models, facilitating the creation of adaptable hybrid retrieval frameworks. Specifically, Search-Adaptor introduces an adaptor function f^d that modifies the original query and corpus embeddings (qe_i^d, ce_j^d) into new embeddings ($\hat{qe}_i^d = f^d(qe_i^d), \hat{ce}_j^d = f^d(ce_j^d)$), where the new similarity score $\hat{s}_{ij}^d = \text{Sim}(\hat{qe}_i^d, \hat{ce}_j^d)$ provides a better estimate of query-corpus relevance. The adaptor function f^d is designed to be trained using a labeled set $R = \{(q_i, c_j, y_{ij})\}_{i=1:N, j=1:M}$. We denote the entire loss function[1] of Search-Adaptor as $\mathcal{L}_{SA}$, which optimizes the adaptor function f^d.

3.3 Sparse retriever tuning

Sparse retrievers, for similarity search using sparse vector representations of text, function by matching keywords between queries and documents. Higher relevance scores are assigned when there are more shared keywords, especially those that are less common in the overall document set. This is typically achieved employing sparse encoder models (E^s) such as BM25, which convert queries and corpus text into sparse embeddings ($qe_i^s = E^s(q_i), ce_j^s = E^s(c_j)$) for comparison (i.e., $s_{ij}^s = \text{Sim}(qe_i^s, ce_j^s)$). Corpus conversion can be done in advance, while queries need to be converted at inference time.

To enhance sparse embedding retrieval, we introduce a novel technique that refines the process using a set of known relevant query-corpus pairs. This involves introducing sparse embedding adaptor models (f^s) that transform the original query and corpus sparse embeddings into customized embeddings ($\hat{qe}_i^s = f^s(qe_i^s), \hat{ce}_j^s = f^s(ce_j^s)$). The new similarity scores with the tuned sparse retriever are calculated as:

$$\hat{s}_{ij}^s = \text{Sim}(qe_i^s, ce_j^s) + \text{Sim}(f^s(qe_i^s), f^s(ce_j^s)). \quad (1)$$

We include the original similarity score s_{ij}^s so that the adaptor function f^s focuses on learning the difference from the original scores for improved retrieval performance.

[1] Further details on $\mathcal{L}_{SA}$ can be found in the original Search-Adaptor paper (Yoon et al., 2023).

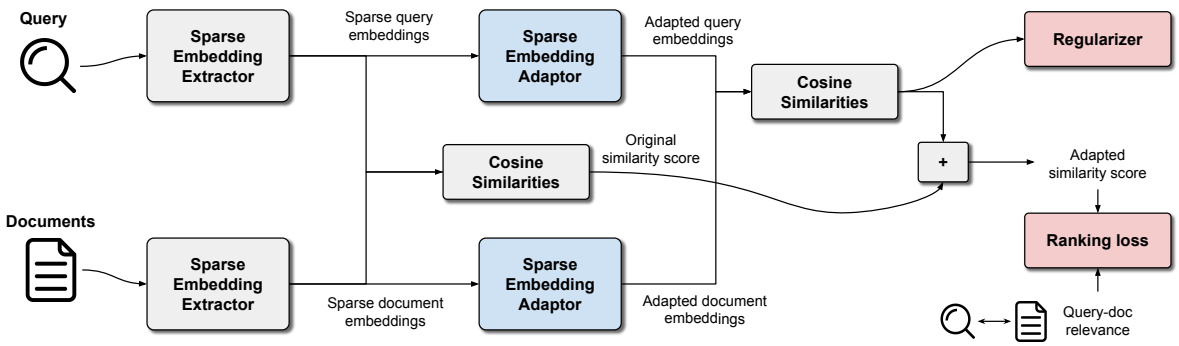

Figure 2: Proposed methods for customization of sparse retrievers. Non-trainable components are shown in grey, while trainable ones are in blue. We utilize two types of losses: ranking loss for incorporating supervised feedback, and a regularization loss to prevent overfitting.

Training the sparse adaptor f^s involves two loss functions. The first term is a regularization loss that prevents excessive modification of the original sparse embedding scores:

$$\mathcal{L}^s_{Reg} = \sum_i \sum_j \text{Sim}(f^s(qe^s_i), qe^s_i)). \quad (2)$$

The second term is a ranking loss that penalizes incorrect ranking compared to the ground truth:

$$\mathcal{L}^s_{Rank} = \sum_i \sum_j \sum_k I(y_{ij} > y_{ik}) \cdot (y_{ij} - y_{ik}) \cdot \log(1 + e^{(\hat{s}^s_{ik} - \hat{s}^s_{ij})}). \quad (3)$$

The sparse adaptor f^s is trained by combining these ranking and regularization losses:

$$\min_{f^s} \mathcal{L}_{Rank}(f^s) + \alpha \mathcal{L}_{Reg}(f^s), \quad (4)$$

where $\alpha \geq 0$ is a hyperparameter that can be determined based on a validation set. Fig. 2 illustrates the entire training procedure of sparse retrievers.

3.4 Aggregation function tuning

The aggregation function fuses two relevance scores, one derived from the dense retriever (s^d_{ij} or $\hat{s}^d_{ij}$) and another from the sparse retriever (s^s_{ij} or $\hat{s}^s_{ij}$). Since these scores might not be on the same scale, simply combining them could disproportionately favor one over the other. In scenarios lacking prior relevance information, a commonly-used solution is Reciprocal Rank Fusion (RRF) (Cormack et al., 2009), defined as:

$$\text{RRF}(s^d, s^s) = \frac{1}{k + \text{rank}(s^d)} + \frac{1}{k + \text{rank}(s^s)}, \quad (5)$$

where k is hyperparameter (often set as 60 in (Cormack et al., 2009)). RRF effectively normalizes the scores by converting them to rankings and then combines these rankings, providing a robust aggregation method in zero-shot settings.

When relevant query-document pairs are available, we can further optimize the aggregation function to enhance the overall performance of the proposed hybrid retrieval framework. We introduce a parameterized aggregation function f^a that takes two scores (s^d_{ij}, s^s_{ij}) as inputs and produces a single aggregated score $s_{ij} = f^a(s^d_{ij}, s^s_{ij})$. With customized scores $(\hat{s}^d_{ij}, \hat{s}^s_{ij})$, the output becomes $\hat{s}_{ij} = f^a(\hat{s}^d_{ij}, \hat{s}^s_{ij})$.

To train the proposed aggregation function, we utilize the same ranking loss as in sparse retriever tuning:

$$\mathcal{L}^a_{Rank} = \sum_i \sum_j \sum_k I(y_{ij} > y_{ik}) \cdot (y_{ij} - y_{ik}) \cdot \log(1 + e^{(\hat{s}_{ik} - \hat{s}_{ij})}). \quad (6)$$

The aggregation function f^a is then trained by minimizing this ranking loss:

$$\min_{f^a} \mathcal{L}^a_{Rank}(f^a). \quad (7)$$

3.5 Joint training

While training all three components of the hybrid retrieval framework simultaneously is feasible, we propose a multi-stage approach for increased stability: *Pre-training stage*: Initially, the dense and sparse embedding adaptors are trained independently and then their parameters are frozen. *Aggregation function training stage*: Next, the aggregation function is trained using the outputs of these

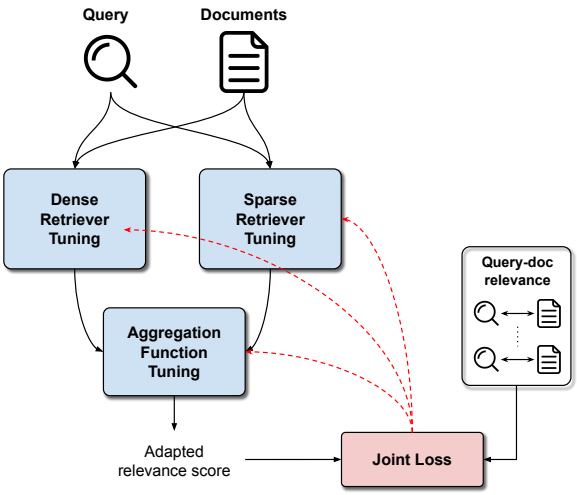

Figure 3: **HyReT** joint training framework. The entire hybrid retrieval pipeline is trained holistically to minimize the loss defined in Eq. (8).

frozen adaptors. *Joint fine-tuning stage*: Finally, the entire system undergoes fine-tuning with a reduced learning rate to achieve optimal performance:

$$\min_{f^d, f^s, f^a} \mathcal{L}^a_{Rank}(f^d, f^s, f^a). \quad (8)$$

We utilize shallow multi-layer perceptrons (MLPs) for the dense and sparse adaptor functions, as well as the aggregation function, to prevent overfitting. The aggregation function (f^a) is a 3-layer MLP with (5, 5, 1) hidden dimensions and a sigmoid activation, resulting in under 50 trainable parameters. The dense adaptor (f^d) is a 1-layer MLP with input dimension matching the hidden dimension, while the sparse adaptor (f^s) is a 1-layer MLP with 64 hidden dimensions and SELU activation. Importantly, f^d includes a skip connection to facilitate learning the difference from the original embeddings. All hyperparameters are optimized using a separate validation set.

4 Experiments

We thoroughly evaluate **HyReT**'s performance across a variety of scenarios and datasets. Overall, our results show that **HyReT** effectively integrates with different types of dense and sparse retrievers, consistently and significantly outperforming the baselines on both English and multilingual datasets.

4.1 Experimental settings

Our study employs two sets of retrieval datasets: English datasets from BEIR (Thakur et al., 2021) and non-English datasets from MIRACL (Zhang et al., 2023). We utilize existing train/test splits when available, otherwise creating disjoint splits. BEIR datasets are divided equally for training and testing, while MIRACL datasets are split as 80% for training and 20% for testing. We evaluate performance using nDCG@10 and further split the training set into 80% training and 20% validation for model selection and hyperparameter optimization. Information regarding data statistics and the specifics of hyper-parameters can be found in Appendix A.1 and A.2, respectively.

To showcase the versatility of **HyReT**, we experiment with different combinations of dense and sparse retrievers. For dense retrieval, we employ either Google Gecko text-embedding-004[2], OpenAI text-embedding-3-small[3], or GTE-Large (Li et al., 2023). For sparse retrieval, we use either BM25 (Robertson et al., 2009) or SPLADE (Formal et al., 2021b). Primary analysis utilizes the SPLADE-v2-max version, further investigations using an alternative SPLADE version are detailed in Appendix B.2. Unless otherwise specified, we default to Google Gecko embedding as the dense retriever and BM25 as the sparse retriever in our experiments.

Details regarding the efficient computations employed in **HyReT** can be found in Appendix A.3 which demonstrates the low computational cost and complexity associated with **HyReT**. For instance, the total training time for all three models is less than 2 hours using 4 V100 GPUs, and the number of additional parameters introduced by **HyReT** is moderate (less than 3 million).

For baselines, we utilize dense, sparse, and hybrid retrieval methods in their original, unmodified form (zero-shot). This means that no customization or fine-tuning is applied to these baseline methods. To showcase the effectiveness of the joint optimization strategy employed by our **HyReT** model, we also compare its performance against partially customized versions of these retrievers.

In our training procedure, we train three separate adapters for each dataset, ensuring that there is no mixing of training data across different datasets. This approach ensures that each adapter is specifically tailored to the characteristics of its corresponding dataset. **HyReT** primarily focuses on en-

[2] https://cloud.google.com/vertex-ai/generative-ai/docs/embeddings/get-text-embeddings

[3] https://platform.openai.com/docs/guides/embeddings

hancing the performance of the baseline zero-shot hybrid search by leveraging available in-domain data. This emphasis on in-domain improvement motivated us to concentrate on in-domain evaluation in our experiments.

4.2 Sparse retriever customization

Datasets	BM25		SPLADE	
	Base	Custom	Base	Custom
NFCorpus	0.3195	**0.3378**	0.3211	**0.3368**
SciFact	0.6729	**0.7986**	0.6361	**0.7368**
Arguana	0.4811	**0.5001**	0.4495	**0.4634**
SciDocs	**0.1594**	0.1591	**0.1599**	0.1569
FiQA	0.1444	**0.2321**	0.2928	**0.3212**
Trec-covid	0.5553	**0.5781**	0.6788	**0.7284**
Touche	0.3757	**0.4945**	0.2523	**0.2721**
Average	0.3869	**0.4429**	0.3986	**0.4308**

Table 2: nDCG@10 performance comparison of sparse retriever customization with two base sparse retrievers (BM25, SPLADE) using BEIR datasets.

While previous work have explored customizing dense retrievers, the customization of sparse retrievers remains less studied, leaving its potential impact unclear. In our first experiment, we focus on investigating the effect of sparse retriever customization across multiple BEIR datasets. To showcase the versatility of **HyReT**, we apply it to two distinct sparse retrievers and evaluate its effectiveness. This experiment highlights the effectiveness of our novel contribution of sparse retriever customization.

Table 2 presents our results, demonstrating consistent and significant improvements in the performance of both base sparse retrievers (BM25 and SPLADE) when **HyReT** is applied. This underscores the potential for performance enhancement through customization not only for dense retrievers, but for sparse retrievers as well.

4.3 Aggregation function tuning

The choice of aggregation method in hybrid retrieval, which combines the relevance scores from dense and sparse retrievers, is a critical factor for optimal performance. Our second experiment focuses on comparing various aggregation methods for both customized and base dense and sparse embeddings, highlighting the significance of our novel aggregation function tuning. In zero-shot setup, we evaluate combining base embeddings and combining customized embeddings. Additionally, we assess the impact of customizing the aggregation function in both cases.

With base embeddings				
Datasets	Mean	Max	RRF	Custom
NFCorpus	0.3688	0.3956	0.3768	**0.4080**
SciFact	0.7406	0.7203	0.7491	**0.7657**
Arguana	0.5562	0.5481	0.5733	**0.6156**
SciDocs	0.2098	**0.2225**	0.2079	0.2162
FiQA	0.4189	0.5458	0.4254	**0.5846**
Trec-covid	0.7032	0.5553	0.7837	**0.8162**
Touche	0.3853	0.3757	**0.4028**	0.3950
Average	0.4833	0.4805	0.5027	**0.5430**
With customized embeddings				
NFCorpus	0.3420	0.3423	**0.4169**	0.3990
SciFact	0.8392	0.8119	**0.8431**	0.8385
Arguana	0.5908	0.6065	0.6290	**0.6947**
SciDocs	0.2138	0.2375	0.2156	**0.2418**
FiQA	0.4189	0.5458	0.4254	**0.5946**
Trec-covid	0.7214	0.5781	0.8415	**0.8549**
Touche	0.5089	0.4945	0.4860	**0.5184**
Average	0.5193	0.5167	0.5511	**0.5917**

Table 3: Comparison of retrieval performance (nDCG@10) across diverse aggregation function tuning approaches, including base embeddings and customized embeddings.

Table 3 presents a comparison of our customized aggregation function tuning with three standard zero-shot aggregation methods: mean, max, and RRF. RRF outperforms both mean and max across multiple datasets, attributed to its better handling of the unnormalized nature of dense and sparse retriever scores, while direct summation or maximization being less reliable. Our customized solution consistently and significantly outperforms all zero-shot aggregation methods, regardless of whether the input scores are from base or customized retrievers. This emphasizes on the importance of customization at every stage of hybrid retrieval.

4.4 Hybrid retrieval tuning

We evaluate the overall performance of **HyReT** by integrating all three tuning components: dense retriever tuning, sparse retriever tuning, and aggregation function tuning. Building upon our earlier experiments, we now evaluate the cumulative impact of these improvements.

Table 4 reveals the significant gains achieved through dense retriever customization compared to the base dense retriever. This consistent improvement across all customization levels demonstrates the widespread benefit of tuning each component of **HyReT**. The most noteworthy result is the perfor-

Datasets	Base Dense Retriever	Base Sparse Retriever	RRF between Two Base Retrievers	Customized Dense Retriever	Customized Sparse Retriever	**HyReT**
NFCorpus	**0.4021**	0.3195	0.4080	0.4014	0.3378	0.3990
SciFact	0.7532	0.6729	0.7491	0.8164	0.7986	**0.8385**
Arguana	0.6127	0.4811	0.5733	0.6839	0.5001	**0.6947**
SciDocs	0.2216	0.1594	0.2079	0.2374	0.1591	**0.2418**
FiQA	0.5822	0.1444	0.4254	0.5922	0.2322	**0.5946**
Trec-covid	0.7871	0.5553	0.8162	0.8157	0.5781	**0.8549**
Touche	0.2627	0.3757	0.3950	0.3035	0.4945	**0.5184**
Average	0.5174	0.3869	0.5107	0.5501	0.4429	**0.5917**

Table 4: Comparison of retrieval performance (nDCG@10) between **HyReT** and base dense/sparse embeddings, as well as partially customized models, using BEIR datasets.

Datasets	Base Dense Retriever	Base Sparse Retriever	RRF between Two Base Retrievers	Customized Dense Retriever	Customized Sparse Retriever	**HyReT**
Yoruba (yo)	0.7643	0.5044	0.7552	0.7793	0.6493	**0.7951**
Swahili (sw)	0.6746	0.3505	0.5635	0.6913	0.5809	**0.7188**
Bengali (bn)	0.6683	0.3154	0.5758	0.6860	0.4323	**0.7313**
Hindi (hi)	0.5530	0.2856	0.5292	0.5589	0.3162	**0.6052**
Telugu (te)	0.7531	0.2073	0.5594	0.7755	0.4363	**0.7995**
Thai (th)	0.6677	0.1277	0.5178	0.6833	0.2316	**0.7163**
Average	0.6802	0.2985	0.5835	0.6957	0.4411	**0.7277**

Table 5: Comparison of retrieval performance (nDCG@10) using **HyReT** against base dense and sparse embeddings, as well as partially customized models, on MIRACL multilingual datasets.

mance of the fully-tuned hybrid retrieval (rightmost column). It demonstrates substantial improvements over all solutions without customization, as well as partially-customized solutions (e.g., those with only tuned dense or sparse retrievers). Optimizing our hybrid retrieval system resulted in a significant performance boost. We achieved a 15.9% improvement over the best zero-shot methods and a 7.6% improvement compared to systems that were partially customized. Specifically, we saw an average increase of over 0.08 in nDCG@10 across the BEIR benchmark. On some datasets, like FiQA, the improvement over the best zero-shot approach was even greater, exceeding 0.16 nDCG@10.

5 Discussions

We present additional experimental analyses to demonstrate the flexibility of **HyReT** and delve into the factors that contribute to its performance.

5.1 Multilingual datasets

To showcase the wide applicability of **HyReT**, we assess the performance on multilingual datasets beyond English as well. We use eight diverse MIRACL datasets for this, with the same experimental setup as before, but replacing the English version of Google's dense retriever with its multilingual counterpart textembedding-gecko-multilingual@002. As for BM25, we utilize the *BERT-Base, Multilingual Cased* vocabulary.

Table 5 demonstrates that **HyReT** consistently and significantly outperforms both zero-shot and partially-customized models. Notably, the proposed customized sparse retrievers also achieve significant improvements over the base sparse retrievers. These emphasize the versatility of **HyReT**, confirming its efficacy across diverse datasets and languages.

5.2 Integrating diverse dense and sparse retrievers

We demonstrate the flexibility of **HyReT** by showing its compatibility with various combinations of dense and sparse retrievers. To achieve this, we present experimental results using different combinations than those used in Section 4. In addition to the Google Gecko and BM25 combination, we test Google Gecko with SPLADE[4], OpenAI text-embedding-3-small with BM25, and GTE-Large (Li et al., 2023) with BM25. For the zero-shot hybrid retrieval results in these combinations, we

[4] https://pypi.org/project/sparsembed/

Datasets	Google + SPLADE		OpenAI + BM25		GTE-Large + BM25	
	Hybrid Retrieval	**HyReT**	Hybrid Retrieval	**HyReT**	Hybrid Retrieval	**HyReT**
NFCorpus	0.3721	**0.4073**	0.3647	**0.3958**	0.3608	**0.3957**
SciFact	0.7093	**0.8256**	0.7407	**0.8465**	0.7467	**0.8254**
Arguana	0.5692	**0.6190**	0.5379	**0.6324**	0.5643	**0.6157**
SciDocs	0.2076	**0.2202**	0.2024	**0.2326**	0.2164	**0.2519**
FiQA	0.4569	**0.5991**	0.3595	**0.4513**	0.3609	**0.4503**
Trec-covid	0.8152	**0.8359**	0.7967	**0.8304**	0.8030	**0.8292**
Touche	0.2983	**0.3472**	0.3540	**0.5036**	0.3366	**0.4010**
Average	0.4898	**0.5506**	0.4794	**0.5561**	0.4841	**0.5385**

Table 6: Comparison of retrieval performance (nDCG@10) using **HyReT** with various combinations of dense and sparse retrievers.

Datasets	Without Sparse Retriever Tuning	Without Dense Retriever Tuning	Without Aggregation Function Tuning	Without Joint Fine Tuning	Without Separate Pretraining	**HyReT**
NFCorpus	0.4009	0.3827	**0.4169**	0.3399	0.3668	0.3990
SciFact	0.8136	0.7972	0.8331	0.8326	0.7522	**0.8385**
Arguana	0.6886	0.6419	0.6290	0.6734	0.6838	**0.6947**
SciDocs	0.2415	0.2106	0.2156	0.2378	0.1681	**0.2418**
FiQA	0.5929	0.5837	0.4254	0.5922	0.4948	**0.5946**
Trec-covid	0.8145	0.7796	0.8415	0.8487	0.8381	**0.8549**
Touche	0.3936	0.5033	0.4860	0.5154	0.4924	**0.5184**
Average	0.5637	0.5570	0.5496	0.5771	0.5423	**0.5917**

Table 7: Comparison of retrieval performance (nDCG@10) using **HyReT** against modified versions.

employ RRF as the aggregation function.

Table 6 shows the consistently improved performance of **HyReT** compared to zero-shot hybrid retrieval, regardless of the specific dense and sparse retriever combination. Moreover, the level of improvement remains similar across different combinations, clearly indicating the model-agnostic nature of **HyReT**. More experimental results with GTE-Large and LoRA tuning (Hu et al., 2021) can be found in the Appendix. B.3.

5.3 Ablation studies

To gain deeper insights into the factors driving the performance improvements of **HyReT**, we conduct ablation studies by systematically removing key components and observing the resulting performance degradations. These reveal the individual contribution of each component to the overall performance. We evaluate five modified versions of **HyReT**, as detailed in Table 7. For this analysis, we utilize Google Gecko as the base dense retriever and BM25 as the base sparse retriever.

Table 7 demonstrates that each of the five key components of **HyReT** contributes to the overall performance enhancement. Notably, independently pre-training the dense retriever, sparse retriever, and aggregation function significantly improves robustness and retrieval performance. While joint fine-tuning provides further gains, the initial separate pre-training plays a crucial role for the overall improvements.

6 Conclusions

Dense and sparse retrievers offer complementary benefits, and hybrid retrieval aims to leverage the strengths of both. While customization of dense retrievers has been extensively studied, the customization of sparse and hybrid retrieval had been under-researched. We propose, **HyReT**, a unified solution that enables customization of the entire hybrid retrieval pipeline with some labeled data. We introduce novel methodological approaches for effective customization of sparse retrievers and the aggregation function. Through extensive experiments across wide range of datasets, we demonstrate that **HyReT** significantly outperforms zero-shot methods and partially-customized solutions. As most state-of-the-art search systems utilize hybrid retrieval, **HyReT** is well positioned to further enhance their retrieval performance.

7 Limitations and Future Work

The proposed version of **HyReT** necessitates relevant query-corpus pairs for training, which may not be easily accessible in all scenarios. Moreover, the scarcity of labeled data can render model selection and hyperparameter tuning less robust. Consequently, developing fully-unsupervised approaches that solely rely on corpus data would be beneficial and can streamline the customization process. Another promising direction for future research would be investigating the ensemble of multiple dense and sparse retrievers within the **HyReT** framework. This could potentially enhance performance by leveraging the strengths of various retrievers.

References

Akiko Aizawa. 2003. An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, 39(1):45–65.

Ravish Bhagdev, Sam Chapman, Fabio Ciravegna, Vitaveska Lanfranchi, and Daniela Petrelli. 2008. Hybrid search: Effectively combining keywords and semantic searches. In *The Semantic Web: Research and Applications: 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008 Proceedings 5*, pages 554–568. Springer.

Danqi Chen and Wen-tau Yih. 2020. Open-domain question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 34–37, Online. Association for Computational Linguistics.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*.

Gordon V Cormack, Charles LA Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 758–759.

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

Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021b. Splade: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2288–2292.

Luyu Gao and Jamie Callan. 2021. Condenser: a pre-training architecture for dense retrieval. *arXiv preprint arXiv:2104.08253*.

Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. 2021. Complement lexical retrieval model with semantic residual embeddings. In *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28–April 1, 2021, Proceedings, Part I 43*, pages 146–160. Springer.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? *Preprint*, arXiv:2310.06770.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.

Jinhyuk Lee, Zhuyun Dai, Xiaoqi Ren, Blair Chen, Daniel Cer, Jeremy R Cole, Kai Hui, Michael Boratko, Rajvi Kapadia, Wen Ding, et al. 2024. Gecko: Versatile text embeddings distilled from large language models. *arXiv preprint arXiv:2403.20327*.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

Gabriel de Souza P Moreira, Radek Osmulski, Mengyao Xu, Ronay Ak, Benedikt Schifferer, and Even Oldridge. 2024. Nv-retriever: Improving text embedding models with effective hard-negative mining. *arXiv preprint arXiv:2407.15831*.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, et al. 2022. Text and code embeddings by contrastive pre-training. *arXiv preprint arXiv:2201.10005*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Tao Shen, Xiubo Geng, Chongyang Tao, Can Xu, Xiaolong Huang, Binxing Jiao, Linjun Yang, and Daxin Jiang. 2023. Lexmae: Lexicon-bottlenecked pretraining for large-scale retrieval. *Preprint*, arXiv:2208.14754.

Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han yu Wang, Haisu Liu, Quan Shi, Zachary S. Siegel, Michael Tang, Ruoxi Sun, Jinsung Yoon, Sercan O. Arik, Danqi Chen, and Tao Yu. 2024. Bright: A realistic and challenging benchmark for reasoning-intensive retrieval. *Preprint*, arXiv:2407.12883.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Jinsung Yoon, Sercan O Arik, Yanfei Chen, and Tomas Pfister. 2023. Search-adaptor: Text embedding customization for information retrieval. *arXiv preprint arXiv:2310.08750*.

Jinsung Yoon, Raj Sinha, Sercan O Arik, and Tomas Pfister. 2024. Matryoshka-adaptor: Unsupervised and supervised tuning for smaller embedding dimensions. *Preprint*, arXiv:2407.20243.

Xinyu Zhang, Nandan Thakur, Odunayo Ogundepo, Ehsan Kamalloo, David Alfonso-Hermelo, Xiaoguang Li, Qun Liu, Mehdi Rezagholizadeh, and Jimmy Lin. 2023. Miracl: A multilingual retrieval dataset covering 18 diverse languages. *Transactions of the Association for Computational Linguistics*, 11:1114–1131.

Xueliang Zhao, Wei Wu, Can Xu, Chongyang Tao, Dongyan Zhao, and Rui Yan. 2020. Knowledge-grounded dialogue generation with pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3377–3390, Online. Association for Computational Linguistics.

A Appendix

A.1 Data Statistics

BEIR dataset				MIRACL dataset			
Sub-data	Number of train pairs	Number of test pairs	Number of corpus	Sub-data	Number of train pairs	Number of test pairs	Number of corpus
NFCorpus	110575	12334	3633	Yoruba (yo)	959	229	49043
SciFact	919	339	5183	Swahili (sw)	9359	5092	131924
Arguana	703	703	8674	Bengali (bn)	16754	4206	297265
SciDocs	14972	14956	25657	Hindi (hi)	11668	3494	506264
FiQA	14166	1706	57638	Telugu (te)	18608	1606	518079
Trec-Covid	35460	30876	171332	Thai (th)	21293	7573	542166
Touche	1077	1137	382545				

Table 8: Statistical overview of BEIR and MIRACL datasets, organized by the number of corpus documents within each dataset.

A.2 Hyper-parameters

We provide a summary of the fixed hyper-parameters used for **HyReT** across all experiments.

Hyper-parameters	Fixed values
Sparse retriever customization coefficient (α)	0.1
Batch size for training	128
Maximum number of training iterations (Pre-training & aggregation function training)	2000
Maximum number of training iterations (Joint fine-tuning)	500
Patience for early stopping (Pre-training & aggregation function training)	500
Patience for early stopping (Joint fine-tuning)	200
Learning rates (Pre-training & aggregation function training)	0.001
Learning rates (Joint fine-tuning)	0.0001
Optimizer	Adam

Table 9: Hyper-parameters used to train **HyReT** approach in all experiments.

A.3 Computational details

All experiments were conducted on 4 NVIDIA V100 GPUs with 16GB of memory. Datasets containing fewer than 100,000 corpus samples were processed in less than 60 minutes, while datasets with up to 2 million corpus samples were processed in less than two hours.

The extra components **HyReT** adds to zero-shot Hybrid Retrieval are three simple multi-layer perceptrons (applied to the dense retriever, sparse retriever, and aggregation function) with less than 3M parameters. This means the extra computation needed during inference, compared to zero-shot Hybrid Retrieval, is negligible.

A.4 Qualitative analyses

In this section, we present illustrative query examples where our **HyReT** model successfully identifies relevant information, while dense or sparse retrievers fail to do so.

Dataset	Query (Orange highlights tokens that appear infrequently.)	Dense retriever	Sparse retriever	**HyReT**
NFCorpus	(i) What do you think of Dr. Jenkins' take on paleolithic diets? (ii) Counteracting the Effects of Dioxins Through Diet	X	O	O
	(iii) Stopping Heart Disease in Childhood (iv) Preventing Brain Loss with B Vitamins?	O	X	O
SciFact	(i) FoxO3a activation in neuronal death is mediated by reactive oxygen species (ROS). (ii) Neutrophil extracellular traps (NETs) are released by ANCA-stimulated neutrophils.	X	O	O
	(iii) Obesity decreases life quality. (iv) The one-child policy has been successful in lowering population growth.	O	X	O
Arguana	(i) An ICC enforcement is a necessity if there is to be international criminal justice. (ii) Overcomes prejudice Affirmative action is required to overcome existing prejudice in universities' admissions procedures.	X	O	O
	(iii) A UN standing army would be ideally suited to respond to contemporary crises. (iv) Free movement will provide benefits for productivity.	O	X	O
SciDocs	(i) Train-O-Matic: Large-Scale Supervised Word Sense Disambiguation in Multiple Languages without Manual Training Data. (ii) Self-Adhesive and Capacitive Carbon Nanotube-Based Electrode to Record Electroencephalograph Signals From the Hairy Scalp.	X	O	O
	(iii) An Attack Graph-Based Probabilistic Security Metric. (iv) Volume of signaling traffic reaching cellular networks from mobile phones.	O	X	O
FiQA	(i) Is it possible to create a self-managed superannuation fund to act as a mortage offset? (Australia) (ii) How and Should I Invest (As a college 18 year old with minimal living expenses)?	X	O	O
	(iii) How are various types of income taxed differently in the USA? (iv) What is considered high or low when talking about volume?	O	X	O
Treccovid	(i) do individuals who recover from COVID-19 show sufficient immune response, including antibody levels and T-cell mediated immunity, to prevent re-infection? (ii) what evidence is there for dexamethasone as a treatment for COVID-19?	X	O	O
	(iii) What new public datasets are available related to COVID-19? (iv) what causes death from Covid-19?	O	X	O
Touche	(i) Should marijuana be a medical option? (ii) Should the penny stay in circulation?	X	O	O
	(iii) Should corporal punishment be used in schools? (iv) Should felons who have completed their sentence be allowed to vote?	O	X	O

Table 10: Illustrative examples where dense or sparse retrievers fail to identify relevant information, yet **HyReT** successfully does so (using BEIR datasets).

B Additional experiments

B.1 Experiments with smaller labeled data

To further emphasize the robustness of **HyReT** when faced with limited training data, we conducted experiments using reduced training set sizes. Specifically, on the SciFact and Arguana datasets, we decreased the number of training samples to 200, 300, and 400 (from the original 919 and 703, respectively).

Method	Number of labeled data	SciFact	Arguana
Hybrid Retrieval	-	0.7491	0.5733
HyReT	200	0.7970	0.6515
	300	0.8003	0.6749
	400	0.8120	0.6870
	Full	0.8285	0.6947

Table 11: Retrieval performance of **HyReT** compared with the zero-shot baseline (Hybrid Retrieval). We utilize smaller number of training samples to verify the impact of **HyReT** even with the small number of labeled data.

Even with these smaller training sets, **HyReT** consistently outperformed the baseline hybrid search method (without tuning). We observed notable improvements in nDCG@10 across all training set sizes. For SciFact, the gains were 0.0478 (with 200 samples), 0.0512 (with 300 samples), and 0.0629 (with 400 samples), while for Arguana, the improvements were even more pronounced at 0.0782 (with 200 samples), 0.1016 (with 300 samples), and 0.1131 (with 400 samples). These findings strongly demonstrate **HyReT**'s ability to effectively optimize hybrid search in low-resource scenarios, highlighting its practical value in real-world applications where labeled data may be scarce.

B.2 Experiments with different SPLADE version

In the main manuscript, we utilize the splade-v2-max version of SPLADE. In this subsection, we have conducted additional experiments with another version, splade-cocondenser-ensembledistil[5].

Datasets	Google + SPLADE-v2-max		Google + SPLADE-new	
	Hybrid Retrieval	**HyReT**	Hybrid Retrieval	**HyReT**
NFCorpus	0.3721	**0.4073**	0.3774	**0.4213**
SciFact	0.7093	**0.8256**	0.7473	**0.8319**
Arguana	0.5692	**0.6190**	0.6317	**0.7028**
SciDocs	0.2076	**0.2202**	0.2154	**0.2266**
FiQA	0.4569	**0.5991**	0.4919	**0.5922**
Trec-covid	0.8152	**0.8359**	0.7734	**0.8293**
Touche	0.2983	**0.3472**	0.2911	**0.3502**
Average	0.4898	**0.5506**	0.5040	**0.5649**

Table 12: Comparison of retrieval performance (nDCG@10) using **HyReT** with SPLADE-new version (SPLADE-cocondenser-ensembledistil).

As can be seen in Table. 12, the improvement margins are consistent with splade-v2-max represented in Table. 6. It is important to emphasize that **HyReT**'s advantage lies in its flexibility to adapt to any dense and sparse retriever model. Regardless of the specific sparse retriever's performance characteristics, **HyReT**, as a canonical framework, consistently delivers further improvements.

[5] https://huggingface.co/naver/splade-cocondenser-ensembledistil

B.3 Experiments with GTE-Large with LoRA tuning

To further examine the impact of customizing both BM25 and the aggregation function, we conducted experiments with LoRA (Hu et al., 2021) tuning on the GTE-large model for each dataset. LoRA tuning is limited to open-source models. Therefore, we can't use it to fine-tune proprietary models like Google's gecko or OpenAI's text-embedding-3. We used the same training data as **HyReT** to establish a benchmark for tuning GTE-Large with LoRA while keeping the sparse retrievers and aggregation function fixed.

Datasets	GTE-Large + BM25		
	Zero-shot Hybrid Retrieval	LoRA-tuned Hybrid Retrieval	**HyReT**
NFCorpus	0.3608	0.3729	**0.3957**
SciFact	0.7467	0.7672	**0.8254**
Arguana	0.5643	0.5739	**0.6157**
SciDocs	0.2164	0.2196	**0.2519**
FiQA	0.3609	0.3635	**0.4503**
Trec-covid	0.8030	0.7958	**0.8292**
Touche	0.3366	0.3767	**0.4010**
Average	0.4841	0.4957	**0.5385**

Table 13: Comparison of retrieval performance (nDCG@10) using **HyReT** with two baselines: (i) zero-shot hybrid retrieval (GTE-Large + BM25 + RRF), (ii) LoRA-tuned hybrid retrieval (LoRA tuned GTE-Large + BM25 + RRF).

Our results demonstrate the importance of holistic optimization in hybrid search. Specifically, on 7 BEIR datasets, GTE-large (with LoRA) combined with BM25 (non-customized) and RRF aggregation (non-customized) achieved a nDCG@10 of 0.4957. In contrast, **HyReT** (with GTE-large) attained a significantly higher nDCG@10 of 0.5385.

This substantial performance gap (0.0428 nDCG@10) can be largely attributed to the effectiveness of **HyReT**'s customized sparse retriever, which significantly outperforms the non-customized BM25, as shown in Table 2. It is also worth noting that RRF aggregation may not be optimal when there are large performance differences between the individual retrieval models. These findings underscore the importance of jointly customizing all three components (dense retriever, sparse retriever, and aggregation function) within a unified framework, as exemplified by **HyReT**.