ROUTERRETRIEVER: Exploring the Benefits of Routing over Multiple Expert Embedding Models

Hyunji Lee$^{\kappa *}$ Luca Soldaini$^{\alpha}$ Arman Cohan$^{\gamma,\alpha}$ Minjoon Seo$^{\kappa}$ Kyle Lo$^{\alpha}$

$^{\kappa}$ KAIST AI $^{\alpha}$ Allen Institute for AI $^{\gamma}$ Yale University

hyunji.amy.lee@kaist.ac.kr {lucas, kylel}@allenai.org

Abstract

Information retrieval methods often rely on a single embedding model trained on large, general-domain datasets like MSMARCO. While this approach can produce a retriever with reasonable overall performance, models trained on domain-specific data often yield better results within their respective domains. While prior work in information retrieval has tackled this through multi-task training, the topic of combining multiple domain-specific expert retrievers remains unexplored, despite its popularity in language model generation. In this work, we introduce ROUTERRETRIEVER, a retrieval model that leverages multiple domain-specific experts along with a routing mechanism to select the most appropriate expert for each query. It is lightweight and allows easy addition or removal of experts without additional training. Evaluation on the BEIR benchmark demonstrates that ROUTERRETRIEVER outperforms both MSMARCO-trained (+2.1 absolute nDCG@10) and multi-task trained (+3.2) models. This is achieved by employing our routing mechanism, which surpasses other routing techniques (+1.8 on average) commonly used in language modeling. Furthermore, the benefit generalizes well to other datasets, even in the absence of a specific expert on the dataset. To our knowledge, ROUTERRETRIEVER is the first work to demonstrate the advantages of using multiple domain-specific expert embedding models with effective routing over a single, general-purpose embedding model in retrieval tasks[1].

Introduction

While a single embedding model trained on large-scale general-domain datasets like MSMARCO (Campos et al. 2016) often performs well, research shows that models trained on domain-specific datasets, even if smaller, can achieve superior results within those domains (Izacard et al. 2021; Bonifacio et al. 2022). Moreover, finetuning on MSMARCO after pretraining with contrastive learning can sometimes degrade performance on specific datasets (Wang et al. 2023; Lee et al. 2023). To improve embedding models for domain-specific datasets, previous studies have explored approaches such as data construction (Wang et al. 2021; Ma et al. 2020) and domain adaptation methods (Xin et al. 2021; Fang et al. 2024). However, less attention has been paid to leveraging multiple expert embedding models and routing among them to select the most suitable one during inference.

In this work, we introduce ROUTERRETRIEVER, a retrieval model that leverages *multiple* domain-specific experts with a routing mechanism to select the most suitable expert for each instance. For each domain, we train gates (experts), and during inference, the model determines the most relevant expert by computing the average similarity between the query and a set of pilot embeddings representing each expert, selecting the expert with the highest similarity score. ROUTERRETRIEVER is lightweight, as it only requires the training of parameter-efficient LoRA module (Hu et al. 2021) for each expert, resulting in a minimal increase in parameters. Additionally, ROUTERRETRIEVER offers significant flexibility: unlike a single model that requires retraining when domains are added or removed, ROUTERRETRIEVER simply adds or removes experts without the need for further training.

Evaluation on the BEIR benchmark (Thakur et al. 2021) with various combinations of experts highlights the benefits of having multiple expert embedding models with a routing mechanism compared to using a single embedding model. When keeping the total number of training datasets constant, ROUTERRETRIEVER consisted of only domain-specific experts without an MSMARCO expert outperforms both a model trained on the same dataset in a multi-task manner and a model trained with MSMARCO. Also, adding domain-specific experts tends to improve performance even when an expert trained on a large-scale general-domain dataset like MSMARCO is already present, suggesting that, despite the capabilities of a general-domain experts, domain-specific experts provide additional benefits, underscoring their importance. Moreover, ROUTERRETRIEVER consistently improves performance as new experts are added, whereas multi-task training tends to show performance degradation when a certain number of domains are included. This indicates the advantage of having separate experts for each domain and using a routing mechanism to select among them. Notably, the benefits of ROUTERRETRIEVER generalize not only to datasets that have corresponding experts but also to additional datasets without specific experts.

We further explore the factors behind these performance benefits. First, ROUTERRETRIEVER consistently shows im-

[*] Work performed during internship at AI2.

[1]Code in https://github.com/amy-hyunji/RouterRetriever

proved performance with the addition of more experts (gates), suggesting that broader domain coverage by experts enhances retrieval accuracy. This trend holds even in an oracle setting, where the gate that maximizes performance is always selected. Notable, adding a new expert for a different domain yields greater performance gains than adding additional experts within the same domain. Second, we observe that parametric knowledge influences embedding extraction. This observation supports the idea that training with domain-specific knowledge improves the quality of embedding extraction of the domain. Last, the performance difference between an instance-level oracle (which routes each instance to its best expert) and a dataset-level oracle (which routes queries to the expert with the highest average performance for the dataset) suggests that queries may benefit from a knowledge of other domains, supporting the effectiveness of our routing technique. Our results point to potential research opportunities in improving routing techniques among multiple expert retrievers, a direction that leads to the development of a retriever system that performs well across both general and domain-specific datasets.

Related Works

Domain Specific Retriever There exists substantial research on retrieval models that aim to improve performance on domain-specific tasks. One approach focuses on dataset augmentation. As domain-specific training datasets are often unavailable and can be costly to construct, researchers have developed methods that either train models in an unsupervised manner (Lee, Chang, and Toutanova 2019; Gao, Yao, and Chen 2021; Gao and Callan 2021) or fine-tune models on pseudo-queries generated for domain-specific datasets (Bonifacio et al. 2022; Ma et al. 2020; Wang et al. 2021). Another approach is developing domain-specific embeddings. A common approach is training in a multi-task manner over domain-specific datasets (Lin et al. 2023; Wang et al. 2021). Recent works have aimed to improve domain-specific retrievers by developing instruction-following retrieval models (Asai et al. 2022; Weller et al. 2024; Oh et al. 2024; Su et al. 2022; Wang et al. 2023); instruction contains such domain knowledge. Another example is Fang et al. (2024) which trains a soft token for domain-specific knowledge. While these methods also aim to extract good representative embeddings for the input text, these methods rely on a *single* embedding model and produce domain-specific embeddings by additionally including domain-specific knowledge (e.g., appended as instructions) to the input. ROUTERRETRIEVER differs from these prior methods by allowing for the employment of *multiple* embedding models where rather than providing the domain knowledge to the input, added to the model as parametric knowledge to produce the domain representative embeddings.

Routing Techniques Various works have focused on developing domain-specific experts and routing mechanisms to improve general performance in generation tasks. One approach simultaneously trains experts (gates) and the routing mechanism (Sukhbaatar et al. 2024; Muqeeth et al. 2024). Another line of work includes post-hoc techniques

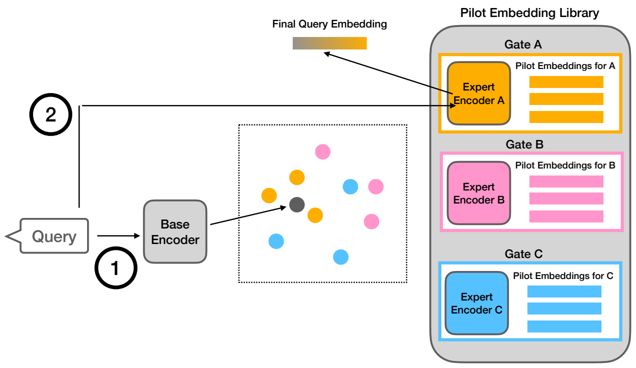

Figure 1: ROUTERRETRIEVER: ① Given a query, we first extract its embedding using a base encoder. We then calculate an average similarity between the query embedding (black dot) and the pilot embeddings for each gate (orange dots for Gate A, red dots for Gate B, and blue dots for Gate C). The gate with the highest average similarity (Gate A in this case) is selected. ② The final query embedding is then produced by passing the query to Expert Encoder A, which consists of the base encoder combined with Gate A, the selected expert gate (LoRA).

that do not require additional training for routing. Some approaches use the model itself as the knowledge source by training it on domain-specific knowledge (Feng et al. 2023), incorporate domain-specific knowledge in the token space (Belofsky 2023; Shen et al. 2024), or select the most relevant source from a sampled training dataset of each domain (Ye et al. 2022; Jang et al. 2023). Routing techniques have also been investigated for improving generation quality in retrieval-augmented generation tasks; Mallen et al. (2022) explores routing to decide whether to utilize external knowledge and Jeong et al. (2024) focuses on routing to choose among different retrieval approaches. However, there has been less emphasis on applying these techniques to information retrieval tasks. In this work, we investigate the benefits of leveraging multiple domain-specific experts and routing mechanisms in information retrieval, contrasting this approach with the traditional methods of using a single embedding model trained on a general-domain dataset or multi-task training across various domains. Additionally, we find that simply adapting routing techniques from generation tasks to information retrieval does not yield high performance, underscoring the importance of developing routing techniques tailored specifically for information retrieval.

Router Retriever

In this section, we introduce ROUTERRETRIEVER, a retrieval model composed of a base retrieval model and *multiple* domain-specific experts (gates). As shown in Figure 1, for a given input query, ① the most appropriate embedding is selected using a routing mechanism. Then, ② the query embedding is generated by passing the query through the selected gate alongside the base encoder.

In the offline time, we train the **experts (gates)** with

Algorithm 1: Constructing Pilot Embedding Library

```
Require: Domain-specific training datasets $D_1, \ldots, D_T$, gates $\mathcal{G} = \{g_1, \ldots, g_T\}$
 1: Initialize empty set $\mathcal{P} = \{\}$ for the pilot embedding library
 2: for each dataset $D_i$ in $\{D_1, \ldots, D_T\}$ do
 3:   Initialize an empty list $\mathcal{L}_i \leftarrow [\,]$
 4:   for each instance $x_j$ in $D_i$ do
 5:     $g_{\max}(x_j) \leftarrow \arg\max_{g_l \in \mathcal{G}} \text{Perf.}(g_l, x_j)$ // Find the gate with maximum performance $g_{\max}$ for instance $x_j$
 6:     Add pair $(x_j, g_{\max})$ to $\mathcal{L}_i$
 7:   end for
 8:   for each gate $g_m$ in $\mathcal{G}$ do
 9:     $Group_m \leftarrow \{x_j \mid g_{\max} = g_m \text{ for } (x_j, g_{\max}) \text{ in } \mathcal{L}_i\}$ // Group all instances $x_j$ for which $g_m$ is the maximum performing gate
10:     if $Group_m$ is not empty then
11:       $\mathbf{E} \leftarrow \text{BaseEncoder}(Group_m)$ // Extract embeddings using the base encoder
12:       $\mathbf{c}_m \leftarrow \text{k-means}(\mathbf{E}, k = 1)$ // Compute the centroid embedding by clustering cluster size 1, which is the pilot embedding
13:       if $g_m$ exists in $\mathcal{P}$ then
14:         Append $\mathbf{c}_m$ to the list associated with $g_m$ in $\mathcal{P}$ ($\mathcal{P}[g_m]$)
15:       else
16:         Add a new entry $\{g_m : [\mathbf{c}_m]\}$ to $\mathcal{P}$
17:       end if
18:     end if
19:   end for
20: end for
21: Output: Pilot embeddings library $\mathcal{P}$
```

domain-specific training datasets and construct a **pilot embedding library**. This library contains pairs of pilot embeddings for each domain along with the corresponding expert trained on that domain. Please note that this process is performed only once. During inference (online time), when given an input query, a **routing mechanism** determines the appropriate expert. We calculate the similarity score between the input query embedding and the pilot embeddings in the pilot embedding library, and then choose the expert with the highest average similarity score.

We use Contriever (Izacard et al. 2021) as the base encoder and train parameter-efficient LoRA (Hu et al. 2021) for each domain as the gate for that domain keeping the model lightweight. For example, in the case of Figure 1, ROUTERRETRIEVER includes a base encoder with three gates (experts): Gate A, Gate B, and Gate C, and the Expert Encoder A is composed of the base encoder with Gate A (LoRA trained on a dataset from domain A) added. This approach allows for the flexible addition or removal of domain-specific gates, enabling various gate combinations without requiring further training for the routing mechanism.

Experts (Gates) For each domain D_i, where $i = 1, \ldots, T$ and T is the total number of domains, we train a separate expert (gate) g_i using the corresponding domain dataset. After the training step, we have a total of T different gates, $\mathcal{G} = \{g_1, g_2, \ldots, g_T\}$, with each gate g_i specialized for a specific domain.

Pilot Embedding Library Given a domain-specific training dataset $D_i = \{x_1, \ldots, x_k\}$ where x_j is an instance in D_i, we perform inference using all gates $\mathcal{G}$ to identify which gate provides the most suitable representative embedding for each instance (line 4-7 in Alg. 1). For each instance x_j, we select $g_{\max}$, the gate that demonstrates the highest performance, defined as $g_{\max}(x_i) = \arg\max_{g_j \in \mathcal{G}} \text{Performance}(g_j, x_i)$. This process produces pairs $(x_j, g_{\max})$ for all instances in the dataset D_i.

Next, we group these pairs by $g_{\max}$, constructing T groups, one for each domain. Then for each group, we perform k-means clustering with cluster size 1 to get the pilot embedding (line 8-19 in Alg. 1). In specific, with the constructed pairs $(x_j, g_{\max})$, we group them by the ones that have the same $g_{\max}$, $Group_m$, which contains list of instances x_j with same gate as the max gate. This results in T groups, one for each domain ($m = 1, \cdots, T$). If the $Group_m$ is not empty, we first extract all embeddings for instances in the group with the base encoder (BaseModel). We then apply k-means clustering () to these embeddings with a cluster size of one. The centroid of this cluster $\mathbf{c}_m$ is taken as the pilot embedding for the domain. This results in one pilot embedding per group, yielding a maximum of T pilot embeddings for the training dataset D_i. Each of these embeddings is associated with a different gate, representing the most suitable one for that domain. Please note that since when $Group_m$ is empty, we do not extract pilot embedding for the empty group (cluster), thereby the number of pilot embeddings for the training dataset could be less than T.

By repeating this process across all domain-specific training datasets $D_1, \ldots, D_T$, we obtain T pilot embeddings for each gate, one from each domain-specific training dataset (repeating line 3-19 in Alg. 1 for all training dataset $D_1 \cdots D_T$). Consequently, the pilot embeddings contains a maximum of T^2 pilot embeddings, with each of the T domain-specific training datasets contributing up to T pilot embeddings.

For example, consider a scenario with three experts, each trained on one of the following datasets: SciFact, FiQA-2018, and HotpotQA. To construct the pilot embedding library, we first perform inference on all training instances used to train the SciFact expert across all three experts to determine which expert produces the most suitable embedding (line 4-7 in Alg. 1). The chosen gate can be any of the three experts. Next, we group all training instances from the SciFact dataset according to the expert that achieved the highest performance for each instance, resulting in up to three groups: instances where the SciFact expert, FIQA-2018 expert, or HotpotQA expert performed best. For each group, we apply k-means clustering with $k = 1$ to compute the centroid, which serves as the pilot embedding for that group. This pilot embedding is added to the pilot embedding library, with the corresponding expert as the key. For example, if the centroid is extracted from the group where the HotpotQA expert performed best, the HotpotQA expert is considered the most suitable expert, even if the instances are from the SciFact dataset. This process adds three pilot embeddings to the library, one for each expert (lines 8-19 in Alg. 1). We repeat this process for all domains (the example on top is for SciFact and we repeat the process for FIQA-2018 and HotpotQA), ultimately creating a total of a maximum of nine pilot embeddings in the library, with three pilot embeddings associated with each expert.

Routing Mechanism When given an input query, we calculate the similarity between the query embedding extracted from the base encoder and the T^2 pilot embeddings in the pilot embedding library. We then average the similarity scores for T pilot embeddings associated with the same gate, resulting in a mean similarity score for each gate. The gate corresponding to the highest mean similarity score is selected as the most suitable embedding model.

Experimental Setup

Baselines We compare the performance of ROUTERRETRIEVER with when training on the same dataset in a multi-task manner (Multi-Task) and training on a large-scale general-domain dataset MSMARCO (MSMARCO-Trained). Additionally, following previous works (Muqeeth et al. 2024; Jang et al. 2023), we evaluate performance using two oracle settings: Best Individual and Oracle. The Best Individual setting is a dataset-level oracle that routes all queries in a dataset to the expert with the highest average performance for that dataset, while the Oracle setting is an instance-level oracle that routes each individual instance to its best-performing expert.

We also conduct experiments with various other routing techniques commonly used in language modeling tasks; ExpertClassifierRouter (Shen et al. 2024), ClassificationHeadRouter (Muqeeth et al. 2024), and DatasetRouter (Ye et al. 2022; Jang et al. 2023). ExpertClassifierRouter employs a binary classifier for each gate to calculate the probability of that gate being selected. The gate with the highest probability is chosen for the final selection. ClassificationHeadRouter uses a single classifier layer to determine the appropriate expert for each instance. DatasetRouter is the most similar to ROUTERRETRIEVER, as it selects the gate by retrieving the instance with the highest similarity score. However, there are two key differences: ROUTERRETRIEVER uses the predicted label, whereas DatasetRouter relies on the original dataset label. Also, ROUTERRETRIEVER incorporates a clustering step to group instances, while DatasetRouter randomly samples 100 instances from the training dataset. Further details of the baselines and training methods for each are provided in the supplementary materials.

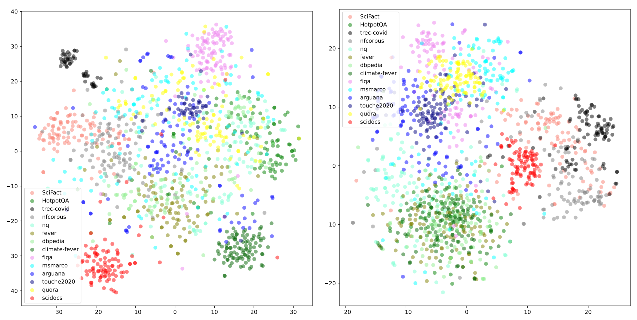

Figure 2: TSNE visualization of contriever embeddings for queries (left) and contexts (right) when sampled 100 instances from each dataset. Datasets with the same domain are in similar colors.

Dataset We used datasets in BEIR benchmark (Thakur et al. 2021), which includes 14 datasets across 6 domains: Bio-Medical, Wikipedia, Finance, Misc., Quora, and Scientific[2]. To train domain-specific gates, we utilize the training sets provided by BEIR. Due to the limited number of datasets with available training sets, we also employed generated queries provided by BEIR[3]. The models were evaluated using the test sets. We categorize the datasets in the Misc. domain as separate general domains, Wikipedia as a general domain, and Bio-Medical, Finance, Quora, and Scientific as domain-specific datasets based on how broadly each instance is distributed. As illustrated in Figure 2, which shows the embeddings extracted from the pre-trained Contriever model (our base model), datasets in the Misc. domain are often widely dispersed even within the same domain. Although the Wikipedia datasets are generally close to others within the same domain, they also exhibit a broad spread. In contrast, datasets from the Bio-Medical, Finance, Quora, and Scientific domains tend to be more compact and closely clustered.

Hyperparameters We use the pre-trained Contriever (Izacard et al. 2021) as our base encoder and train gates (LoRA) according to the settings in Lee et al. (2023), with a rank of 8, an alpha of 32 per gate, thereby training approximately 0.5% of the parameters (about 1M

[2]Details of datasets and domains in supplementary
[3]https://huggingface.co/BeIR

	Misc		Wiki	Bio	Science	Quora	Finance	
	AR	MS	HO	NF	SF	QU	FI	Avg
MSMARCO-Trained	37.2	**25.7**	57.6	31.7	67.2	**84.1**	28.8	47.5
Multi-Task	36.9	22.4	52.1	32.9	69.4	82.0	28.9	46.4
RouterRetriever	**38.6**	23.0	**59.9**	**33.4**	**77.6**	83.8	**30.8**	**49.6**
RouterRetriever (w/o MS expert)	39.5	22.2	59.5	33.4	76.0	83.6	30.5	49.3
Best Individual	40.2	25.7	59.9	34.4	79.8	84.5	32.2	50.9
Oracle	48.5	34.5	66.6	39.0	85.4	89.9	39.6	57.6

Table 1: RouterRetriever consistently outperforms both the MSMARCO-Trained and Multi-Task models in terms of nDCG@10. Even without using the MSMARCO expert (RouterRetriever w/o MS expert), thus maintaining the same total number of training datasets as the MSMARCO-Trained, RouterRetriever achieves superior performance. This result highlights that the performance improvement comes from having diverse domain-specific experts rather than simply the amount of training data. Additionally, RouterRetriever performs comparably to the Best Individual model.

parameters) per gate. For training, we adopt the few-shot hyperparameters from Izacard et al. (2021): a learning rate of 1e-4, a batch size of 256 with in-batch negatives, and a maximum of 500 epochs with early stopping. Gates are applied only to the query encoder, keeping the context encoder frozen, as our focus is on understanding the impact of routing by query instances, thereby eliminating the influence of routing on the context encoder. We also include the results of applying gates to the context encoder in the supplementary materials.

Experimental Results & Discussions

Overall Performance

Table 1 shows the performance of RouterRetriever compared to baseline models using seven domain-specific gates (AR, MS, HO, NF, SF, QU, and FI)[4]. RouterRetriever outperforms the MSMARCO-trained model, indicating that even with a large-scale general-domain training dataset, incorporating additional domain-specific gates further enhances performance. Also, when keeping the training dataset the same when comparing RouterRetriever to the Multi-Task model, which is trained with the same training datasets, RouterRetriever consistently shows higher performance. Moreover, RouterRetriever (w/o MS expert), which excludes the MSMARCO gate but maintains the same total number of training datasets as the MSMARCO-trained model, still achieves superior performance. These results underscore the importance of having separate embedding models (gates) for each domain and dynamically selecting the most appropriate gate for each query rather than relying on a single model to handle multiple domains. For additional results with different combinations of experts, please refer to the supplementary material.

Affect of Dataset Size when Training Experts

Figure 3 shows the relationship between performance (y-axis) and the number of training samples (x-axis) across

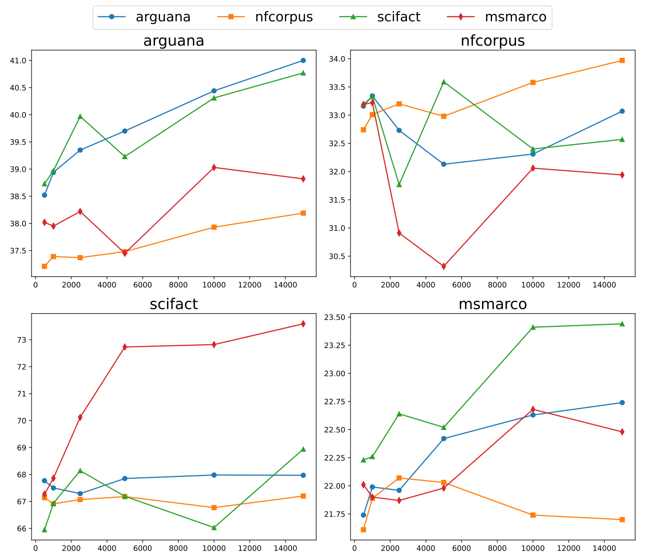

Figure 3: nDCG@10 (y-axis) is plotted against training dataset size (x-axis). Each line color represents the training dataset used, and the title of each plot indicates the evaluation dataset.

various datasets. For in-domain evaluation datasets, performance generally improves as the number of training samples increases. However, in out-of-domain evaluation datasets, simply increasing the number of training samples does not necessarily lead to better performance. When the same number of training samples is used, for in-domain cases, experts consistently achieve the highest performance across all evaluation datasets. Interestingly, for out-of-domain cases, experts perform better when trained on general domains (e.g., Arguana and MSMARCO) compared to domain-specific experts (e.g., SciFact and NFcorpus). We attribute this to the broader coverage and stability of general-domain datasets, as illustrated in Figure 2. These results suggest that while a larger training dataset is generally beneficial for expert in-domain performance, the coverage of the training dataset has a more significant impact on out-of-domain performance.

[4]All gates, except for MSMARCO, are selected to have the smallest training dataset from each domain to ensure that the total number of training dataset is equal to that of MSMARCO when excluding the MS gate.

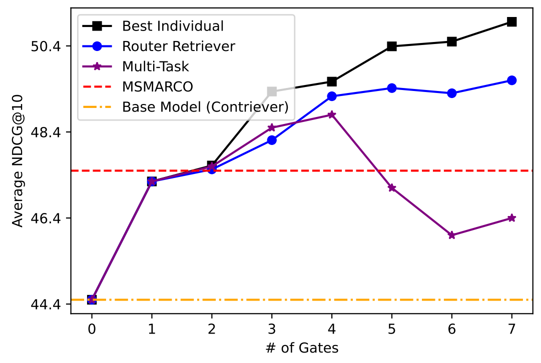

Figure 4: Average nDCG@10 (y-axis) by the number of gates (x-axis) for various models. ROUTERRETRIEVER tend to show improved performance as number of gates increases, outperforming MSMARCO-trained model even with just three gates and showing small gap with Best Individual performance.

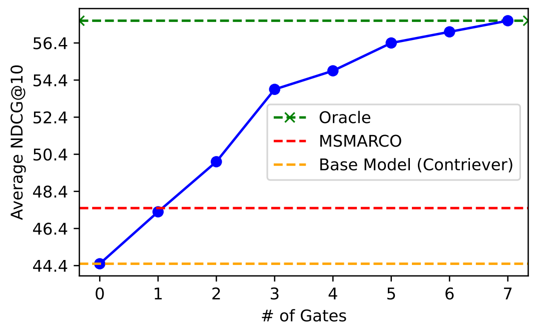

Figure 5: Average oracle nDCG@10 (y-axis) by the number of available gates (x-axis). The improvement rate tend to be higher when adding gates initially and as the number of gates grows, the rate of increase diminshes.

Impact of Number of Gates

Figure 4 shows that adding gates (x-axis) consistently improves the performance of ROUTERRETRIEVER (y-axis). Notably, ROUTERRETRIEVER outperforms the MSMARCO-trained model even with just three gates, indicating that despite not having as diverse or large a training dataset as MSMARCO, the advantage of having multiple embedding models and the ability to select the most suitable one leads to better performance. ROUTERRETRIEVER also shows a small gap with the Best Individual performance which is the in-domain performance for each expert (Oracle performance for dataset-wise). The performance in multi-task training tends to fluctuate as the number of domains (gates) increases. We hypothesize that with a large number of domains, the model struggles to find the optimal embedding for general cases due to the high variance across training datasets.

Figure 5 illustrates the performance when we use 7 gates and increase the number of experts that the model can choose from, selecting the one with the maximum performance for each instance. Oracle represents the performance when the model can route through all 7 gates and choose the best-performing one instance-wise. As the number of gates increases, performance consistently improves. Notably, the rate of improvement is higher when adding gates initially, and as the number of gates grows, the rate of increase diminishes, regardless of the order in which experts are added. We believe this tendency arises because the routing technique tends to be more distracted as more gates are added. Nonetheless, the consistent improvement with additional gates highlights the potential for further enhancement with better routing techniques, emphasizing the importance of investigating these techniques across various expert retrievers. We randomly varied the order and combination of gates in the figure but observed that the trend remained consistent. Details are provided in the supplementary materials.

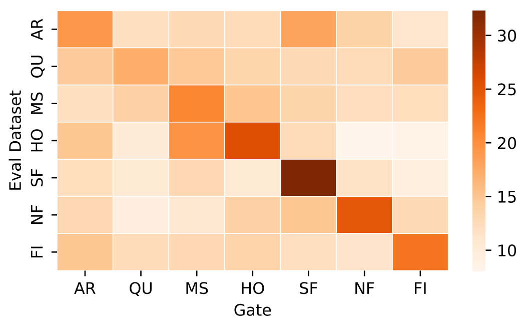

Figure 6: For each evaluation dataset (y-axis), the rate at which gate tends to show the maximum performance (x-axis). While general domain datasets (AR, MS, HO) perform well overall, the domain-specific datasets (SF, NF, FI, QU) demonstrate particularly strong performance within their respective domains, highlighting the importance of having gates for domain-specific tasks.

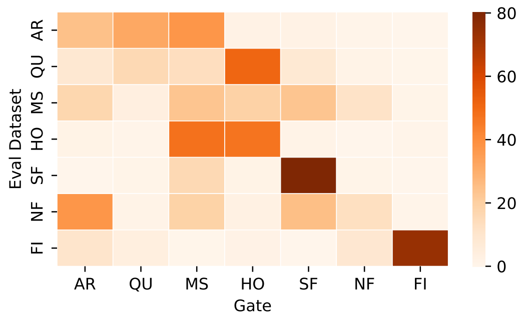

Figure 7: For each evaluation dataset (y-axis), the rate at which gate the router chooses (x-axis).

Maximum Performing Gate Rates and Analysis of Gate Selection

Figure 6 shows the rate at which each gate achieves the highest performance across different evaluation datasets.

	Misc		Wiki	Bio	Science	Quora	Finance	
	AR	MS	HO	NF	SF	QU	FI	Avg
MSMARCO-Trained	37.2	**25.7**	57.6	31.7	67.2	**84.1**	28.8	47.5
ExpertClassifierRouter	37.9	23.8	53.1	31.5	67.1	82.5	29.1	46.4
ClassificationHeadRouter	38.5	22.6	52.8	32.7	69.6	83.4	28.2	46.8
DatasetRouter	37.3	23.6	58.4	33.1	73.4	83.9	29.9	48.5
ROUTERRETRIEVER	**38.6**	23.0	**59.9**	**33.4**	**77.6**	83.8	**30.8**	**49.6**

Table 2: nDCG@10 performance across different routing techniques commonly used in language modeling. ROUTERRETRIEVER consistently achieves the highest performance, highlighting the need for specialized routing techniques in information retrieval, given the distinct objectives and usage compared to language modeling.

For general-domain datasets (AR, MS, HO), the best-performing gate is often distributed across multiple experts. However, for domain-specific datasets (SF, NF, FI, QU), the best performance is typically achieved by the gate trained specifically on that domain. This indicates that while gates generally perform well on general-domain datasets, having a domain-specific expert model is essential for achieving high performance in specialized areas.

Figure 7 shows the rate at which gates are selected by our routing technique for each evaluation dataset. The MS gate is often chosen, likely because, as shown in Figure 2, MS-MARCO instances are broadly distributed, leading to more generalized pilot embeddings. Since the MS gate performs well across all datasets (as seen in Figure 6), this selection seems reasonable. For domain-specific datasets like SF and FI, the routing strongly favors the gate trained on the respective dataset, which we assume is likely because these datasets cluster closely together in Figure 2. We add a detailed error analysis of the routing technique in the supplementary.

Impact of Expert Combinations

We experiment with various combinations of experts to assess their impact on performance. Our findings suggest that broad coverage across domains is critical. Within a single domain, the specific expert chosen does not significantly affect performance as long as it is trained with a sufficient amount of training dataset. Adding an expert from a new domain tends to significantly improve performance while adding additional experts to a domain that already has an expert doesn't yield as much improvement.

Adding domain-specific gates like SciDocs in the Science domain or TREC-COVID in the Bio-Medical domain improves performance for those datasets, with SciDocs increasing from 44.3 to 56.2 and TREC-COVID from 15.1 to 16.1. However, the overall average performance across all datasets remains relatively stable. We also observed that performance tends to improve when using experts trained on larger datasets, consistent with our earlier observation in Figure 3 that expert performance generally increases with the size of the training dataset. Detailed results are provided in the supplementary materials.

	w/ Experts	w/o Experts	Avg
MSMARCO-Trained	47.5	31.6	40.0
Multi-Task	46.4	31.2	38.8
ROUTERRETRIEVER	**49.6**	**31.9**	**40.8**
Best Individual	50.9	34.2	42.6
Oracle	57.6	41.5	49.6

Table 3: ROUTERRETRIEVER show high performance (nDCG@10) for not only datasets with experts but also generalizes to those without experts. Each w/ and w/o Experts contains 7 datasets.

Various Routing Techniques

We experiment with various routing techniques commonly used in language modeling and compared them with our proposed routing mechanism. Results in Table 2 show that the routing technique used in ROUTERRETRIEVER consistently achieves the highest performance. In fact, ClassificationHeadRouter and ExpertClassifierRouter approaches tend to underperform compared to when using a single expert trained solely on MSMARCO (MSMARCO-Trained). DatasetRouter, which is the closest to ROUTERRETRIEVER, tends to show higher performance than MSMARCO-Trained but also consistently shows lower performance than ROUTERRETRIEVER. These results suggest that these routing techniques are not well-suited for information retrieval and may even degrade performance compared to using a single expert. We hypothesize that the differences in the effectiveness of routing techniques between language modeling and information retrieval can be explained from two perspectives. First, in language modeling, experts are often trained to handle distinct tasks, making them easier to differentiate. In contrast, information retrieval involves domain classification, which may be more challenging. Second, in language modeling, routing decisions are often made at the token level, which allows for greater flexibility and reduces the impact of any single choice. However, in information retrieval, where a single representative embedding is required, the choice of expert is made only once per instance, making the process more vulnerable to the routing technique used, and thus requiring greater precision.

General Performance of ROUTERRETRIEVER over various datasets

Table 3 demonstrates that ROUTERRETRIEVER consistently outperforms other baselines that rely on a single general-purpose embedding model[5]. This is evident not only in datasets that have their experts (w/ Experts) but also across various other datasets that do not have their experts (w/o Experts). These findings suggest that the benefits of having multiple experts and routing across them extend well beyond the datasets for which specific experts were trained.

Where does the benefit come from?

We hypothesize that the benefit of having domain-specific gates comes from the model's tendency to be influenced by its parametric knowledge; models trained on domain-specific datasets are likely to have domain-specific knowledge embedded in their parametric space, enabling them to produce more meaningful embeddings related to those domains. To test the hypothesis, we conduct experiments with the dataset from Zhou et al. (2023), which contains both original NQ (Kwiatkowski et al. 2019) contexts that align with the retriever's parametric knowledge and conflicting contexts for each instance. We experiment with RepLlama (Ma et al. 2024) and E5-Mistral (Wang et al. 2023)[6] and found that the retrievers surprisingly for all case prefer contexts that align with their parametric knowledge; they consistently retrieve the original NQ contexts over conflicting contexts[7]. This finding supports our hypothesis that embedding models are influenced by parametric knowledge when extracting embedding thus their knowledge of domain-specific datasets are better able to extract meaningful embeddings relevant to their domain knowledge. Further details are in the supplementary.

Efficiency

ROUTERRETRIEVER achieves high efficiency by using parameter-efficient LoRA gates, which account for only about 0.5% of the parameters per gate. This makes the addition of new gates relatively insignificant in terms of parameter count. In terms of training, it uses the same amount of training data as in a multi-task approach. However, unlike multi-task training, which requires retraining the entire model when adding, removing, or changing domains, ROUTERRETRIEVER allows for these modifications without additional training, as our routing technique is training-free. However, during inference, computing the query embedding involves two forward passes: the first to identify the appropriate gate (routing), and the second to generate the final query embedding. Improving the computation efficiency of this routing technique is a direction for future work.

[5]Detailed numbers of the table are in Supplementary.

[6]We used these two models not Contriever to ensure that the NQ contexts from Zhou et al. (2023) align with their parametric knowledge

[7]We exclude contexts containing an extensive length of contexts (context with table information) as they tend to introduce bias (Thakur et al. 2021)

Conclusion

In this paper, we present ROUTERRETRIEVER, a retrieval model that integrates multiple domain-specific experts with a routing mechanism to extract the most suitable embedding for each query. This approach is both lightweight and flexible, allowing for the addition or removal of experts without additional training. Our experiments demonstrate that it consistently outperforms single embedding models, showcasing the advantages of integrating domain-specific experts. Additionally, it surpasses various widely used routing techniques in language modeling, emphasizing the significance of effective routing for information retrieval tasks. These results highlight the crucial role of domain-specific experts in improving retrieval performance and suggest that combining them with efficient routing techniques can significantly enhance results, potentially approaching oracle performance.

Acknowledgments

We thank Nandan Thakur, Orion Weller, Jiyeon Kim, and Hanseok Oh for helpful discussions and constructive feedback.

References

Asai, A.; Schick, T.; Lewis, P.; Chen, X.; Izacard, G.; Riedel, S.; Hajishirzi, H.; and Yih, W.-t. 2022. Task-aware retrieval with instructions. *arXiv preprint arXiv:2211.09260.*

Belofsky, J. 2023. Token-Level Adaptation of LoRA Adapters for Downstream Task Generalization. In *Proceedings of the 2023 6th Artificial Intelligence and Cloud Computing Conference*, 168–172.

Bondarenko, A.; Fröbe, M.; Beloucif, M.; Gienapp, L.; Ajjour, Y.; Panchenko, A.; Biemann, C.; Stein, B.; Wachsmuth, H.; Potthast, M.; and Hagen, M. 2020. Overview of Touché 2020: Argument Retrieval. In *Conference and Labs of the Evaluation Forum.*

Bonifacio, L.; Abonizio, H.; Fadaee, M.; and Nogueira, R. 2022. Inpars: Data augmentation for information retrieval using large language models. *arXiv preprint arXiv:2202.05144.*

Boteva, V.; Ghalandari, D. G.; Sokolov, A.; and Riezler, S. 2016. A Full-Text Learning to Rank Dataset for Medical Information Retrieval. In *European Conference on Information Retrieval.*

Campos, D. F.; Nguyen, T.; Rosenberg, M.; Song, X.; Gao, J.; Tiwary, S.; Majumder, R.; Deng, L.; and Mitra, B. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *ArXiv*, abs/1611.09268.

Cohan, A.; Feldman, S.; Beltagy, I.; Downey, D.; and Weld, D. S. 2020. SPECTER: Document-level Representation Learning using Citation-informed Transformers. *ArXiv*, abs/2004.07180.

Diggelmann, T.; Boyd-Graber, J. L.; Bulian, J.; Ciaramita, M.; and Leippold, M. 2020. CLIMATE-FEVER: A Dataset for Verification of Real-World Climate Claims. *ArXiv*, abs/2012.00614.

Fang, Y.; Ai, Q.; Zhan, J.; Liu, Y.; Wu, X.; and Cao, Z. 2024. Combining Multiple Supervision for Robust Zero-Shot Dense Retrieval. In *AAAI Conference on Artificial Intelligence*.

Feng, S.; Shi, W.; Bai, Y.; Balachandran, V.; He, T.; and Tsvetkov, Y. 2023. Knowledge Card: Filling LLMs' Knowledge Gaps with Plug-in Specialized Language Models. *arXiv preprint arXiv:2305.09955*.

Gao, L.; and Callan, J. 2021. Condenser: a Pre-training Architecture for Dense Retrieval. In Moens, M.-F.; Huang, X.; Specia, L.; and Yih, S. W.-t., eds., *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Gao, T.; Yao, X.; and Chen, D. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Hasibi, F.; Nikolaev, F.; Xiong, C.; Balog, K.; Bratsberg, S. E.; Kotov, A.; and Callan, J. 2017. DBpedia-Entity v2: A Test Collection for Entity Search. *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Hu, J. E.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; and Chen, W. 2021. LoRA: Low-Rank Adaptation of Large Language Models. *ArXiv*, abs/2106.09685.

Izacard, G.; Caron, M.; Hosseini, L.; Riedel, S.; Bojanowski, P.; Joulin, A.; and Grave, E. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Jang, J.; Kim, S.; Ye, S.; Kim, D.; Logeswaran, L.; Lee, M.; Lee, K.; and Seo, M. 2023. Exploring the benefits of training expert language models over instruction tuning. In *International Conference on Machine Learning*, 14702–14729. PMLR.

Jeong, S.; Baek, J.; Cho, S.; Hwang, S. J.; and Park, J. C. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. *arXiv preprint arXiv:2403.14403*.

Kwiatkowski, T.; Palomaki, J.; Redfield, O.; Collins, M.; Parikh, A. P.; Alberti, C.; Epstein, D.; Polosukhin, I.; Devlin, J.; Lee, K.; Toutanova, K.; Jones, L.; Kelcey, M.; Chang, M.-W.; Dai, A. M.; Uszkoreit, J.; Le, Q. V.; and Petrov, S. 2019. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics*, 7: 453–466.

Lee, H.; Soldaini, L.; Cohan, A.; Seo, M.; and Lo, K. 2023. Back to Basics: A Simple Recipe for Improving Out-of-Domain Retrieval in Dense Encoders. *arXiv preprint arXiv:2311.09765*.

Lee, K.; Chang, M.-W.; and Toutanova, K. 2019. Latent Retrieval for Weakly Supervised Open Domain Question Answering. In *ACL 2019*.

Lin, S.-C.; Asai, A.; Li, M.; Oğuz, B.; Lin, J. J.; Mehdad, Y.; tau Yih, W.; and Chen, X. 2023. How to Train Your DRAGON: Diverse Augmentation Towards Generalizable Dense Retrieval. *ArXiv*, abs/2302.07452.

Ma, J.; Korotkov, I.; Yang, Y.; Hall, K.; and McDonald, R. 2020. Zero-shot neural passage retrieval via domain-targeted synthetic question generation. *arXiv preprint arXiv:2004.14503*.

Ma, X.; Wang, L.; Yang, N.; Wei, F.; and Lin, J. 2024. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2421–2425.

Maia, M.; Handschuh, S.; Freitas, A.; Davis, B.; McDermott, R.; Zarrouk, M.; and Balahur, A. 2018. WWW'18 Open Challenge: Financial Opinion Mining and Question Answering. *Companion Proceedings of the The Web Conference 2018*.

Mallen, A.; Asai, A.; Zhong, V.; Das, R.; Khashabi, D.; and Hajishirzi, H. 2022. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*.

Muqeeth, M.; Liu, H.; Liu, Y.; and Raffel, C. 2024. Learning to route among specialized experts for zero-shot generalization. *arXiv preprint arXiv:2402.05859*.

Oh, H.; Lee, H.; Ye, S.; Shin, H.; Jang, H.; Jun, C.; and Seo, M. 2024. INSTRUCTIR: A Benchmark for Instruction Following of Information Retrieval Models. *arXiv preprint arXiv:2402.14334*.

Roberts, K.; Alam, T.; Bedrick, S.; Demner-Fushman, D.; Lo, K.; Soboroff, I.; Voorhees, E. M.; Wang, L. L.; and Hersh, W. R. 2020. TREC-COVID: rationale and structure of an information retrieval shared task for COVID-19. *Journal of the American Medical Informatics Association : JAMIA*, 27: 1431 – 1436.

Shen, S. Z.; Lang, H.; Wang, B.; Kim, Y.; and Sontag, D. 2024. Learning to decode collaboratively with multiple language models. *arXiv preprint arXiv:2403.03870*.

Su, H.; Shi, W.; Kasai, J.; Wang, Y.; Hu, Y.; Ostendorf, M.; Yih, W.-t.; Smith, N. A.; Zettlemoyer, L.; and Yu, T. 2022. One embedder, any task: Instruction-finetuned text embeddings. *arXiv preprint arXiv:2212.09741*.

Sukhbaatar, S.; Golovneva, O.; Sharma, V.; Xu, H.; Lin, X. V.; Rozière, B.; Kahn, J.; Li, D.; Yih, W.-t.; Weston, J.; et al. 2024. Branch-Train-MiX: Mixing Expert LLMs into a Mixture-of-Experts LLM. *arXiv preprint arXiv:2403.07816*.

Thakur, N.; Reimers, N.; Rücklé, A.; Srivastava, A.; and Gurevych, I. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*.

Thorne, J.; Vlachos, A.; Christodoulopoulos, C.; and Mittal, A. 2018. FEVER: a Large-scale Dataset for Fact Extraction and VERification. *ArXiv*, abs/1803.05355.

Wachsmuth, H.; Syed, S.; and Stein, B. 2018. Retrieval of the Best Counterargument without Prior Topic Knowledge. In *Annual Meeting of the Association for Computational Linguistics*.

Wadden, D.; Lo, K.; Wang, L. L.; Lin, S.; van Zuylen, M.; Cohan, A.; and Hajishirzi, H. 2020. Fact or Fiction: Verifying Scientific Claims. *ArXiv*, abs/2004.14974.

Wang, K.; Thakur, N.; Reimers, N.; and Gurevych, I. 2021. GPL: Generative pseudo labeling for unsupervised domain adaptation of dense retrieval. *arXiv preprint arXiv:2112.07577.*

Wang, L.; Yang, N.; Huang, X.; Yang, L.; Majumder, R.; and Wei, F. 2023. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368.*

Weller, O.; Chang, B.; MacAvaney, S.; Lo, K.; Cohan, A.; Van Durme, B.; Lawrie, D.; and Soldaini, L. 2024. FollowIR: Evaluating and Teaching Information Retrieval Models to Follow Instructions. *arXiv preprint arXiv:2403.15246.*

Xin, J.; Xiong, C.; Srinivasan, A.; Sharma, A.; Jose, D.; and Bennett, P. N. 2021. Zero-shot dense retrieval with momentum adversarial domain invariant representations. *arXiv preprint arXiv:2110.07581.*

Yang, Z.; Qi, P.; Zhang, S.; Bengio, Y.; Cohen, W. W.; Salakhutdinov, R.; and Manning, C. D. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Conference on Empirical Methods in Natural Language Processing*.

Ye, S.; Jang, J.; Kim, D.; Jo, Y.; and Seo, M. 2022. Retrieval of soft prompt enhances zero-shot task generalization. *arXiv preprint arXiv:2210.03029.*

Zhou, W.; Zhang, S.; Poon, H.; and Chen, M. 2023. Context-faithful Prompting for Large Language Models. In *Conference on Empirical Methods in Natural Language Processing*.

Experimental Setup

Baselines

MSMARCO This baseline uses a single MSMARCO gate, which is trained on a large-scale, general-domain dataset without any routing techniques applied.

Multi-Task In this approach, we train a single embedding model on all datasets simultaneously in a multi-task manner. We keep the number of training datasets for each label the same, keeping to the one with the minimum value by sampling.

Best Individual This represents the oracle performance when selecting the single best-performing gate for *each dataset*. For example, if the SciFact gate shows the highest overall performance on the SciDocs evaluation dataset compared to other gates, the performance of the SciFact gate is recorded as the best individual performance for SciDocs.

Oracle This is the oracle performance when selecting the best-performing gate for *each individual instance*. For example, within the SciDocs dataset, certain instances might achieve the highest performance with the SciFact gate, while others might perform better with the MSMARCO gate. This baseline measures the performance when, for each instance, the gate that yields the best result is selected.

ExpertClassifierRouter This routing technique, inspired from Shen et al. (2024), uses a binary classifier for each gate. For each instance, the classifier calculates the probability of selecting or not selecting a specific gate. The gate with the highest probability of being selected is chosen.

To construct the training dataset, we use the predicted label (g_{max}) from the **Pilot Embedding Library**. For each (x_i, g_{max}) pair, we randomly sample instances where the maximum gate differs, which are used to train the "not choosing the gate" label. The dataset is balanced across labels, with the following number of training instances for each dataset: AR (16,108), FI (1070), SF (1,414), NF (892), HO (4,618), QU (4,326), and MS (4,252). Please note that the training datasets only consist of instances where only a single gate shows maximum performance. We then train a binary classifier for each gate to predict whether an instance is likely to achieve the highest performance through that gate.

ClassificationHeadRouter This routing technique, inspired from Muqeeth et al. (2024), uses a classification head where the number of labels corresponds to the number of gates. The gate with the highest predicted probability is selected as the one likely to yield the best performance. To ensure balance, we equalize the number of training instances for each label, matching the dataset with the fewest instances (NFcorpus with 892 instances, other numbers in ExpertClassifierRouter paragraph). AS a result, the total number of training instances is 6,244.

DatasetRouter This routing technique, inspired from Ye et al. (2022); Jang et al. (2023), is the closest baseline to ROUTERRETRIEVER. It samples 100 training instances from each dataset and when given a query, it retrieves the most relevant instances from these samples. The gate trained on the dataset from which the sample originated is then used.

The key differences between DatasetRouter and RouterRetriever are as follows. (1) RouterRetriever uses the predicted label to map an instance to a gate, while DatasetRouter relies on the original dataset label. For example, if a training instance from MSMARCO performs best with the sciFact gate, RouterRetriever will select the Scifact gate for a similar query, whereas DatasetRouter will select the MSMARCO gate. (2) RouterRetriever incorporates a clustering step, grouping similar instances together and using centroid embeddings, rather than treating each instance individually.

Datasets

Stats of Training Dataset Table 4 presents the statistics and details of the datasets in the BEIR benchmark, which we used for training and evaluation. We sampled datasets from Quora to ensure that the number of training instances for AR, HO, NF, SF, FI, and QU matches that of MS.

Examples of Oracle Table 5 shows examples of questions where a gate from a different dataset outperforms the gate trained on the dataset to which the question belongs. We observe that questions related to biology often achieve higher performance with the NFCorpus gate, while those involving scientific knowledge tend to favor the SciFact gate, and questions requiring arguments perform better with the Arguana gate. This pattern suggests that, even within a single dataset, some instances may be more closely aligned with other datasets, likely because the datasets were not labeled or constructed to avoid overlap with existing datasets.

Hyperparameters

We trained the Contriever model (Izacard et al. 2021) using an asymmetric architecture, where the query encoder encodes the query and the context encoder encodes the context. In our experiments, we fine-tuned only the LoRA (Low-Rank Adaptation) parameters of the query encoder, training approximately 1 million parameters per gate (which accounts for 0.5% of the total model parameters). For evaluation, we used the NDCG@10 metric, consistent with previous works (Thakur et al. 2021; Lee et al. 2023), which measures the ranking quality of the top 10 retrieved documents. All results were calculated using the official BEIR evaluation code. The experiments were conducted on 8 or fewer A6000 GPUs (each with 40GB of memory). We utilized checkpoints from all pretrained models available on Huggingface[8]. The experiments were performed over various combinations of gates, with all random seeds set to 10.

When unfreeze context encoder In our main experiments, we focus on scenarios where the context encoder is frozen, and only the LoRA of the query encoder is trainable to isolate the impact of routing on the query encoder alone. However, we observe that the overall performance trend remains similar even when the context encoder is not frozen, with the unfrozen models generally achieving higher performance. Table 6 presents the results when the context encoder is frozen. In these experiments, RouterRetriever consistently outperforms the MSMARCO-trained model and the Multi-Task model.

[8]https://huggingface.co/facebook/contriever

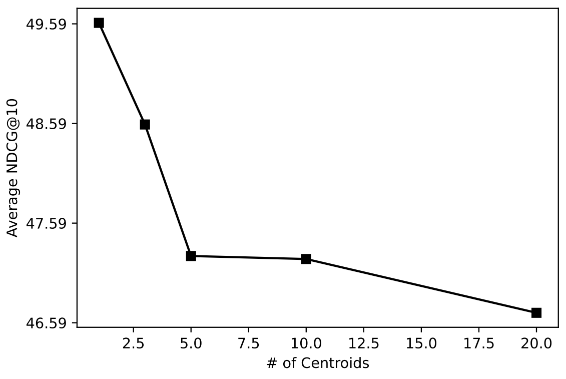

Figure 8: Average NDCG@10 performance (y-axis) as the number of centroid embeddings from k-means clustering increases. The performance tend to decrease with more pilot embeddings, which suggests that when there are too many pilot embeddings, it tends to distract the performance.

Experimental Results & Discussions

Performance of each gates

To analyze the performance trends of each gate, we evaluate them individually without applying any routing techniques in Table 7. The performance generally shows the highest when the evaluation dataset matches the training dataset of the gate. Additionally, the performance gap between matching and non-matching datasets is larger for domain-specific datasets (NF, TR, SD, SF, QU, FI). In contrast, gates trained on general-domain datasets (AR, MS, HO) tend to perform well across a broader range of datasets.

Affect of Number of Pilot Embeddings

We experiment with how the number of pilot embeddings affects performance. In Figure 8, we observe that performance tends to degrade as the number of pilot embeddings increases. We hypothesize that this decline is due to the increased number of pilot embeddings becoming distracting, leading to less effective routing decisions.

Impact of Number of Gates

To investigate the impact of number of gates, we randomly shuffle the gate order and experiment how adding gates tend to affect performance. The order of gates added in Figure 4 and Figure 5 is AR, FI, SF, NF, HO, QU, and MS. We tried various other combinations and could see that the findings are stabilized (Figure 9): (1) performance tend to increase with more gates added and (2) the improvement rate tend to be higher when adding gates initially and as the number of gates grows, the rate of increase diminishes.

Domain	Name	Task	Train (k)	Gen Train (k)	Test	Corpus (k)
Misc.	ArguAna (AR) (Wachsmuth, Syed, and Stein 2018)	Argument Retrieval	-	23	1,406	8.7
	Touche-2020 (TO) (Bondarenko et al. 2020)	Argument Retrieval	-	-	49	382.5
	MSMARCO (MS) (Campos et al. 2016)	Passage-Retrieval	503	-	6,980	8,842
Wikipedia	NaturalQuestions (NQ) (Kwiatkowski et al. 2019)	Question Answering	-	-	3,452	2,681
	HotpotQA (HO) (Yang et al. 2018)	Question Answering	85	-	7,405	5,233
	DBpedia (DB) (Hasibi et al. 2017)	Entity-Retrieval	-	-	400	4,636
	FEVER (FE) (Thorne et al. 2018)	Fact Checking	110	-	6,666	5,417
	Climate-FEVER (CL) (Diggelmann et al. 2020)	Fact Checking	-	-	1,535	5,417
Bio-Medical	TREC-COVID (TR) (Roberts et al. 2020)	Bio-Medical Retrieval	-	432	50	171
	NFCorpus (NF) (Boteva et al. 2016)	Bio-Medical Retrieval	2.6	10.8	323	3.6
Scientific	SCIDOCS (SD) (Cohan et al. 2020)	Citation-Prediction	-	67	1,000	25.7
	SciFact (SF) (Wadden et al. 2020)	Fact Checking	0.8	15.4	300	5.2
Finance	FIQA-2018 (FI) (Maia et al. 2018)	Question Answering	5.5	162	648	57.6
Quora	Quora (QU)	Duplicate-Question Retrieval	-	200	10,000	523

Table 4: Data statics of 14 datasets in BEIR benchmark. Units of the numbers of training dataset (Train), generated training dataset (Gen Train), and corpus are in thousands. For most datasets, training datasets are not provided and for some datasets, we failed to download the generated training dataset.

Table 5: Examples where the dataset label differs from the predicted label based on the highest-performing gate.

Question	**Dataset**	**Max Gate**
APOE4 expression in iPSC-derived neurons results in decreased tau phosphorylation.	SciFact	NFCorpus
which mir regulates the autophagy of cells	SciFact	NFCorpus
what kind of leader should i be as the chief executive	FiQA-2018	Arguana
what is casual dining dining	FiQA-2018	HotpotQA
is it better to be a vegan or vegetarian?	Arguana	SciFact
could we ban animal testing	Arguana	SciFact
why do humans eat meat	Arguana	Quora

	Misc		Wiki	Bio	Science	Quora	Finance	
	AR	MS	HO	NF	SF	QU	FI	Avg
MSMARCO	39.3	**25.3**	57.9	32.2	66.5	**84.3**	28.6	47.7
Multi-Task	38.2	21.9	49.8	32.4	65.1	83.3	26.1	45.3
RouterRetriever	**40.5**	21.0	**61.3**	**32.7**	**68.2**	82.5	**30.0**	**48.0**
Best Individual	41.2	25.3	60.9	32.2	70.3	86.1	32.2	49.8
Oracle	48.2	33.2	68.4	39.0	76.7	90.0	38.6	56.3

Table 6: Performance of RouterRetriever when context encoder is trainable.

Domain	Training Data	AR	TO	MS	CL	DB	NQ	FE	HO	NF	TR	SD	SF	QU	FI
Misc	AR	**40.2**	17.0	22.4	15.7	31.0	26.2	68.9	54.2	32.8	40.3	15.5	67.8	83.7	29.6
	MS	37.2	18.3	**25.7**	16.0	32.7	**29.3**	68.8	57.6	31.7	41.2	14.6	67.2	84.1	28.8
Wiki	HO	38.5	**19.7**	22.4	**17.7**	**36.1**	28.8	67.8	**59.9**	32.2	39.1	16.2	66.0	82.8	27.7
Bio	NF	38.7	17.7	21.8	13.4	28.7	24.0	64.6	46.4	**34.4**	42.1	15.5	66.6	82.5	27.3
	TR	37.0	17.3	22.8	16.2	31.6	26.4	68.1	56.6	33.1	**67.3**	15.7	68.3	83.1	29.1
Science	SD	38.9	18.2	22.8	17.0	32.0	27.3	**70.0**	57.2	33.2	39.6	**16.3**	66.7	84.3	28.4
	SF	37.9	16.5	21.8	16.0	29.4	25.5	68.1	50.2	32.3	28.8	15.1	**79.8**	83.6	25.1
Quora	QU	37.5	19.4	22.6	13.9	29.0	27.4	63.8	49.1	31.1	49.7	14.3	65.7	**84.5**	28.3
Finance	FI	35.1	18.5	22.1	15.4	29.5	25.2	64.6	46.9	32.3	42.7	15.0	64.5	83.4	**32.2**

Table 7: Overall performance when evaluating each gate separately.

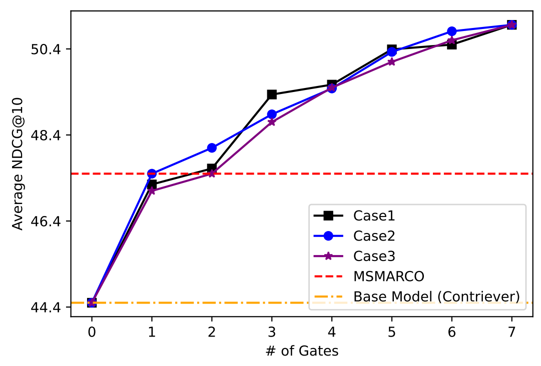

Figure 9: For each evaluation dataset (y-axis), the rate at which gate the router chooses (x-axis). We could see that the trend generalizes with different combination of gates. Case1 is in order of AR, FI, SF, NF, HO, QU, and MS. Case2 is in reverse order of MS, QU, HO, NF, SF, FI, and AR. Case 3 is in order of SF, NF, HO, QU, AR, MS, and FI.

	Misc		Bio	Science	Finance	Avg
	AR	MS	NF	SF	FI	
MSMARCO	37.2	25.7	31.7	67.2	28.8	38.1
Multi-Task	39.4	21.2	28.2	69.2	30.0	37.6
ROUTERRETRIEVER	**40.1**	22.1	32.3	**76.7**	**30.7**	**40.4**
Best Individual	40.2	22.4	34.4	79.8	32.2	41.8
Oracle	47.5	29.3	38.0	84.5	37.4	47.3

Table 8: ROUTERRETRIEVER performance with four gates: AR, NF, SF, FI. Avg is an average performance over the dataset of all gates and MSMARCO.

Detailed numbers by gates

In this section, we show detailed number of performance with different combinations of gates. Table 8 shows performance with AR, NF, SF, FI as gates. Table 9 shows performance with AR, **HO**, NF, SF, FI as gates. Table 10 shows performance with AR, HO, NF, SF, **QU**, FI as gates. Table 11 shows performance with AR, **MS**, HO, NF, SF, QU, FI as gates. Figure 4 shows only with three gates, ROUTERRETRIEVER outperforms the MSMARCO-trained ones thereby in all results, we can see that ROUTERRETRIEVER outperforms the MSMARCO-trained ones and multi-task baselines.

Routing Mechanism Error Analysis

Figure 7 illustrates the rate at which each router selects a gate, while Figure 6 shows the rate at which each gate tends to deliver high performance for the dataset. The discrepancy between these two heatmaps highlights the gap between ROUTERRETRIEVER and the oracle performance. For Arguana, the maximum gate distribution is evenly spread, and the routing tends to follow this distribution closely. For Quora, while the maximum gate rate is high overall, the routing often favors the HotpotQA gate in many cases. For MSMARCO, the gate trained on MSMARCO generally shows high performance, but the routing technique tends to distribute selections across different gates. For HotpotQA, selecting the HotpotQA gate most frequently results in the highest performance, with MSMARCO being the next best option. The routing technique tends to reflect this pattern. For SciFact, choosing the SciFact gate is crucial in both cases. For NFCorpus, selecting the NFCorpus gate is important, yet the routing technique often opts for the Arguana gate in many instances. For FiQA-2018, the best performance is achieved by selecting the FiQA-2018 gate, and the routing technique successfully identifies this gate most of the time.

We specifically investigated why NFCorpus often fails to select the NFCorpus gate and instead tends to choose the Arguana gate. Upon examining the representative embeddings for Arguana, we found that many of them are confused with Arguana embeddings that were extracted from the NFCorpus dataset. These instances originally belong to NFCorpus but show the highest performance with the Arguana gate, leading to their labeling as Arguana. This suggests that instead of completely removing information about the original dataset, incorporating a weighting factor between the two could further improve performance.

	Misc		Wiki	Bio	Science	Finance	Avg
	AR	MS	HO	NF	SF	FI	
MSMARCO	37.2	25.7	57.6	31.7	67.2	**28.8**	41.4
Multi-Task	37.7	22.0	58.6	31.1	69.1	28.4	41.2
ROUTERRETRIEVER	**38.5**	22.3	**59.2**	**33.0**	**72.2**	27.9	**42.2**
Best Individual	40.2	22.4	59.9	34.4	79.8	32.2	44.8
Oracle	47.7	31.5	65.1	38.6	84.8	38.4	51.0

Table 9: ROUTERRETRIEVER performance with five gates: AR, NF, SF, FI, HO. Avg is an average performance over the dataset of all gates and MSMARCO.

Generalization to other datasets

We observe that ROUTERRETRIEVER demonstrates stable performance not only on datasets with corresponding gates but also on those without them. The performance with different numbers of gates is shown in the following tables: Table 13 (4 gates), Table 14 (5 gates), Table 17 (6 gates), Table 18 (7 gates), and Tables 16 and 15 (8 gates).

When using a similar total number of training datasets (Table 17), ROUTERRETRIEVER and the MSMARCO-trained model exhibit comparable generalization performance (both at 31.6). However, ROUTERRETRIEVER achieves higher performance on datasets that have corresponding gates (47.5 for MSMARCO-only vs. 49.3 for ROUTERRETRIEVER). As more gates are added, both generalization ability and performance on datasets with corresponding gates tend to improve (Figure 9).

Where does the benefit come from?

We hypothesize that the advantage of using multiple expert embedding models with routing, rather than a single embedding model, stems from the influence of the training dataset

	Misc		Wiki	Bio	Science	Quora	Finance	
	AR	MS	HO	NF	SF	QU	FI	Avg
MSMARCO	37.2	25.7	57.6	31.7	67.2	**84.1**	28.8	47.5
Multi-Task	35.3	21.5	55.3	32.3	65.3	82.8	29.3	46.0
RouterRetriever	**39.5**	22.2	**59.5**	**33.4**	**76.0**	83.6	**30.5**	**49.3**
Best Individual	40.2	22.6	59.9	34.4	79.8	84.5	32.2	50.5
Oracle	48.0	32.7	65.5	38.8	85.0	89.7	39.2	57.0

Table 10: RouterRetriever performance with six gates: AR, NF, SF, FI, HO, QU. Avg is an average performance over the dataset of all gates and MSMARCO.

	Misc		Wiki	Bio	Science	Quora	Finance	
	AR	MS	HO	NF	SF	QU	FI	Avg
MSMARCO	37.2	**25.7**	**57.6**	31.7	67.2	**84.1**	28.8	47.5
Multi-Task	36.9	22.4	52.1	32.9	69.4	82.0	28.9	46.4
RouterRetriever	**38.6**	23.0	**59.9**	**33.4**	**77.6**	83.8	**30.8**	**49.6**
Best Individual	40.2	25.6	59.9	34.4	79.8	84.5	32.2	50.9
Oracle	48.5	34.5	66.6	39.0	85.4	89.9	39.6	57.6

Table 11: RouterRetriever performance with seven gates: AR, NF, SF, FI, HO, QU, MS

	Misc		Wiki	Bio		Science		Quora	Finance	
	AR	MS	HO	NF	TR	SD	SF	QU	FI	Avg
RouterRetriever	39.5	22.2	59.5	33.4	44.3	15.1	76.0	83.6	30.5	44.9
RouterRetriever (+ TR)	38.4	22.7	59.9	33.3	**56.2**	14.6	77.3	83.9	31.4	46.4
RouterRetriever (+ SD)	38.8	22.9	59.8	32.7	44.7	**16.1**	76.9	84.1	30.3	45.1

Table 12: RouterRetriever performance when adding gates within same domain. The performance tend to improve for the dataset, but for the rest the difference tend to be minor.

	Misc			Wiki					Bio		Science		Quora	Finance	
	AR	TO	MS	CL	DB	NQ	FE	HO	**NF**	TR	SD	**SF**	QU	**FI**	Avg
MSMARCO	37.2	18.3	25.7	16.0	32.7	29.3	68.8	57.6	31.7	41.2	14.6	67.2	84.1	28.8	39.5
Multi-Task	39.4	18.9	21.2	16.1	27.0	23.5	60.0	41.6	28.2	41.6	15.0	69.2	82.2	30.0	36.7
RouterRetriever	40.1	18.4	22.1	15.4	33.6	25.5	67.3	55.3	**32.3**	43.1	15.1	76.7	83.2	30.7	**39.9**
Best Individual	40.2	18.5	22.4	15.7	31.0	26.2	68.9	54.2	34.4	44.6	15.5	79.8	83.7	32.2	40.5
Oracle	47.5	23.8	29.3	19.7	36.5	33.5	76.7	59.4	38.0	52.5	20.0	84.5	88.1	37.4	46.2

Table 13: RouterRetriever performance with four gates: AR, NF, SF, FI. Avg is an average performance over all datasets.

	Misc			Wiki					Bio		Science		Quora	Finance	
	AR	TO	MS	CL	DB	NQ	FE	**HO**	**NF**	TR	SD	**SF**	QU	**FI**	Avg
MSMARCO	37.2	18.3	25.7	16.0	32.7	29.3	68.8	57.6	31.7	41.2	14.6	67.2	84.1	28.8	39.5
Multi-Task	37.7	17.8	22.0	15.4	33.2	26.6	65.5	58.6	31.1	41.6	15.1	69.1	82.7	28.4	39.0
RouterRetriever	38.5	19.8	22.3	17.1	35.4	27.8	67.4	59.2	33.0	43.6	15.9	72.2	82.7	27.9	**40.2**
Best Individual	40.2	19.7	22.4	17.7	36.1	28.8	68.9	59.9	34.4	44.6	16.2	79.8	83.7	32.2	41.8
Oracle	47.7	25.7	31.5	21.3	40.4	37.9	79.5	65.1	38.6	53.1	20.7	84.8	88.7	38.4	48.1

Table 14: RouterRetriever performance with five gates: AR, NF, SF, FI, HO. Avg is an average performance over all datasets.

	Misc			Wiki					Bio		Science		Quora	Finance	
	AR	TO	**MS**	CL	DB	NQ	FE	**HO**	**NF**	TR	**SD**	**SF**	**QU**	**FI**	Avg
MSMARCO	37.2	18.3	25.7	16.0	32.7	29.3	68.8	57.6	31.7	41.2	14.6	67.2	84.1	28.8	39.5
Multi-Task	37.7	17.8	22.0	15.4	33.2	26.6	65.5	58.6	31.1	41.6	15.1	69.1	82.7	28.4	39.0
RouterRetriever	38.8	17.7	22.9	15.2	31.7	27.7	67.5	59.8	32.7	44.7	16.3	76.9	84.1	30.3	**40.4**
Best Individual	40.2	19.7	25.6	17.7	36.1	29.3	70.8	59.9	34.4	49.7	16.2	79.8	84.5	32.2	42.6
Oracle	49.1	27.4	33.8	20.7	42.0	40.8	82.6	66.3	40.2	55.4	18.5	86.1	88.5	33.3	48.9

Table 15: RouterRetriever performance with eight gates: AR, NF, SF, FI, HO, MS, SD, QU. Avg is an average performance over all datasets.

	Misc			Wiki					Bio		Science		Quora	Finance	
	AR	TO	**MS**	CL	DB	NQ	FE	**HO**	**NF**	**TR**	SD	**SF**	**QU**	**FI**	Avg
MSMARCO	37.2	18.3	25.7	16.0	32.7	29.3	68.8	57.6	31.7	41.2	14.6	67.2	84.1	28.8	39.5
Multi-Task	37.5	17.4	23.8	15.5	31.7	26.9	66.9	58.1	34.8	44.5	14.2	68.1	81.0	27.6	39.1
RouterRetriever	38.4	17.6	22.7	15.3	32.1	27.4	67.4	59.9	32.7	56.2	14.6	77.3	83.9	31.4	**41.7**
Best Individual	40.2	19.7	25.6	17.7	36.1	29.3	70.8	59.9	34.4	67.3	16.2	79.8	84.5	32.2	46.9
Oracle	48.6	27.0	35.2	21.5	43.1	41.1	80.3	65.1	41.1	69.1	18.2	84.1	86.4	37.1	49.9

Table 16: RouterRetriever performance with eight gates: AR, NF, SF, FI, HO, MS, TR, QU. Avg is an average performance over all datasets.

	Misc			Wiki					Bio		Science		Quora	Finance	
	AR	TO	MS	CL	DB	NQ	FE	**HO**	**NF**	TR	SD	**SF**	**QU**	**FI**	Avg
MSMARCO	37.2	18.3	25.7	16.0	32.7	29.3	68.8	57.6	31.7	41.2	14.6	67.2	84.1	28.8	39.5
Multi-Task	35.3	17.6	21.5	15.8	31.0	25.9	66.4	55.3	32.3	41.3	14.5	65.3	82.8	29.3	38.2
RouterRetriever	39.5	17.2	22.2	16.0	32.8	27.3	68.1	59.5	33.4	44.3	15.1	76.0	83.6	30.5	**40.4**
Best Individual	40.2	19.7	22.6	17.7	36.1	28.8	68.9	59.9	34.4	49.7	16.2	79.8	84.5	32.2	42.2
Oracle	48.0	26.8	32.7	21.7	40.6	39.3	80.0	65.5	38.8	56.2	21.0	85.0	89.7	39.2	48.9

Table 17: RouterRetriever performance with six gates: AR, NF, SF, FI, HO, QU. Avg is an average performance over all datasets. Number of total training dataset of RouterRetriever, Multi-Task, and MSMARCO-only are the same.

	Misc			Wiki					Bio		Science		Quora	Finance	
	AR	TO	**MS**	CL	DB	NQ	FE	**HO**	**NF**	TR	SD	**SF**	**QU**	**FI**	Avg
MSMARCO	37.2	18.3	25.7	16.0	32.7	29.3	68.8	57.6	31.7	41.2	14.6	67.2	84.1	28.8	38.5
Multi-Task	36.9	17.3	22.4	16.3	32.9	26.7	69.0	52.1	32.9	41.4	14.5	69.4	82.0	28.9	38.8
RouterRetriever	38.6	17.7	23.0	15.2	33.9	27.6	69.2	59.9	33.4	44.9	14.8	77.6	83.8	30.8	**40.7**
Best Individual	40.2	19.7	25.6	17.7	36.1	29.3	70.8	59.9	34.4	49.7	16.2	79.8	84.5	32.2	42.6
Oracle	48.5	27.2	34.5	22.3	41.5	40.8	81.0	66.6	39.0	56.4	21.2	85.4	89.9	39.6	49.6

Table 18: RouterRetriever performance with seven gates: AR, NF, SF, FI, HO, QU, MS. Avg is an average performance over all datasets.

on a model's parametric knowledge, which in turn affects the extracted embeddings. To test this hypothesis, we experimented to determine whether a model tends to prefer contexts that align with its parametric knowledge over those that conflict with it.

We used a dataset released by Zhou et al. (2023), which includes instances where each context either aligns with or conflicts with the model's parametric knowledge. For each instance with 5-6 contexts, we evaluated which context the model chose based on the highest similarity. Interestingly, in all instances[9], the models consistently preferred the context that aligned with their parametric knowledge. This suggests that the internal knowledge of the model influences how embeddings are extracted, and that having domain knowledge embedded in the model's parameters enhances performance.

[9]We excluded contexts containing extensive length of contexts (tables), as they tend to introduce bias (Thakur et al. 2021).