ROUTERRETRIEVER: Routing over a Mixture of Expert Embedding Models

Hyunji Lee[1*], **Luca Soldaini**[2], **Arman Cohan**[2, 3], **Minjoon Seo**[1], **Kyle Lo**[2]

[1]KAIST AI
[2]Allen Institute for AI
[3]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, they often underperform models trained on domain-specific data when testing on their respective domains. Prior work in information retrieval has tackled this through multi-task training, but the idea of routing over a mixture of domain-specific expert retrievers remains unexplored despite the popularity of such ideas in language model generation research. In this work, we introduce ROUTERRETRIEVER, a retrieval model that leverages a mixture of domain-specific experts by using a routing mechanism to select the most appropriate expert for each query. ROUTERRETRIEVER is lightweight and allows easy addition or removal of experts without additional training. Evaluation on the BEIR benchmark demonstrates that ROUTERRETRIEVER outperforms both models trained on MSMARCO (+2.1 absolute nDCG@10) and multi-task models (+3.2). 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. ROUTERRETRIEVER is the first work to demonstrate the advantages of routing over a mixture of domain-specific expert embedding models as an alternative to a single, general-purpose embedding model, especially when retrieving from diverse, specialized domains.

Code — https://github.com/amy-hyunji/RouterRetriever
Extended version — https://arxiv.org/abs/2409.02685

Introduction

Domain-specific retrievers have been shown to outperform general-purpose retrievers for specialized retrieval settings (Izacard et al. 2021; Bonifacio et al. 2022), even in cases where domain-specific training data is only available at a much smaller scale than general-domain datasets like MSMARCO (Campos et al. 2016). Yet, developing and maintaining separate retrieval systems for each specialized retrieval domain can be costly compared to simply maintaining a single general-purpose MSMARCO-trained model. Even employing multi-task training, which combines both MSMARCO as well as domain-specific data, to improve performance of the single model setup can be expensive when considering the need for full model retraining whenever a new target retrieval domain emerges, and may not always preserve performance uniformly across all target domains (Wang et al. 2023; Lee et al. 2023). Research has largely focused on improving the performance of these single model setups through data construction (Wang et al. 2021; Ma et al. 2020) and domain adaptation (Xin et al. 2021; Fang et al. 2024). But less attention has been paid to what can be done if we afford ourselves to use *multiple* specialized retrieval models.

In this work, we introduce ROUTERRETRIEVER, a retrieval model that leverages a mixture of domain-specific experts with a routing mechanism to select the most suitable expert for each instance. ROUTERRETRIEVER consists of a shared base retrieval model and multiple LoRA (Hu et al. 2021) components which serves as experts for specific domains. During training, each expert is trained on a domain-specific dataset while sharing the same frozen base model. Thus the expert component captures domain-specific knowledge and extracts embeddings tailored to the domain. At inference time, as shown in Figure 1, our routing method determines the most appropriate expert by calculating the average similarity between the query and a set of pilot embeddings representing each expert. The expert with the highest similarity score is selected, and the corresponding domain-specific embedding is generated using the chosen expert. ROUTERRETRIEVER is lightweight, as it only requires the training of a parameter-efficient LoRA module for each expert, resulting in a minimal increase in parameters. Additionally, ROUTERRETRIEVER offers significant flexibility: unlike maintaining a single model that requires retraining when domains are added or removed, ROUTERRETRIEVER simply adds or removes experts without the need for further training of the rest of the model.

We demonstrate the effectiveness of ROUTERRETRIEVER on the BEIR benchmark (Thakur et al. 2021) through a series of experiments with various combinations of target domains: First, ROUTERRETRIEVER routing between only domain-specific experts outperforms both training a single model on the same dataset in a multi-task manner and training a single model on MSMARCO only. Second, we ob-

[*]Work performed during internship at Ai2.

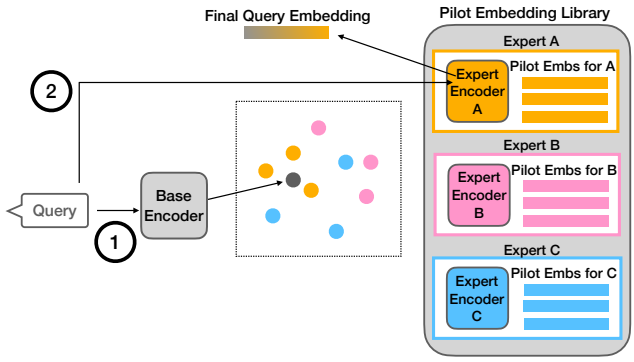


Figure 1: ROUTERRETRIEVER: ① Given a query, we first extract its embedding using a base encoder. We then calculate an average similarity between the query embedding (black dot) and the pilot embeddings for each expert (orange dots for Expert A, red dots for Expert B, and blue dots for Expert C). The expert with the highest average similarity (Expert 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 the selected expert LoRA module.

serve that routing techniques from language modeling research don't necessarily translate to our retrieval setting, which motivates us to introduce a new, specialized routing technique based on embedding similarities. Third, ROUTERRETRIEVER consistently improves performance as we add new experts, whereas multi-task training tends to show performance degradation after a certain number of target domains are included. In fact, even without an expert defined for a target domain, ROUTERRETRIEVER can outperform a single general-purpose model on those unseen domains.

We also conduct analysis to better understand the factors behind these performance benefits. First, to understand data scaling benefits when training each individual expert, we find in-domain performance rapidly increases as we increase training set size, whereas out-of-domain performance often may not see any improvement with scale; hence, the need to use multiple experts. Second, we show that ROUTERRETRIEVER continually improves as the number of experts increases but quickly runs into diminishing returns. These improvements aren't just with respect to overall performance, but also with respect to performance stability across domains. Third, when analyzing routing errors, ROUTERRETRIEVER tends to have a sparser expert selection compared to our instance-level oracle setup.

Related Works

Domain Specific Retriever Substantial research on retrieval models aims 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. 2023b; 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 generate high-quality domain-specific embeddings, they focus on incorporating domain-specific knowledge into the input and processing it with a *single* embedding model. In contrast, ROUTERRETRIEVER employs a mixture of *multiple* embedding models, encoding domain knowledge directly into their parametric representations to produce more effective 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 and the routing mechanism (Sukhbaatar et al. 2024; Muqeeth et al. 2024). Another line of work includes post-hoc techniques 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 use external knowledge and Jeong et al. (2024) focuses on routing to choose among different retrieval approaches. In our work, we observe that directly adapting routing techniques from generation tasks to retrieval does not yield optimal performance. To address this, we introduce a routing technique specifically tailored to retrieval tasks. In information retrieval, (Lin et al. 2023a) introduces a technique that decomposes long and complex queries into sub-queries, which are then routed to specialized expert retrievers. Unlike our work, which employs lightweight components as experts, they rely on separate, individual expert models. Further, while they assign sub-queries to different experts in a rules-based manner, our method processes the entire query and applies dynamic routing.

Router Retriever

In this section, we introduce ROUTERRETRIEVER, a retrieval model composed of a base retrieval model and multiple domain-specific experts. 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 se-

Algorithm 1: Constructing Pilot Embedding Library

Require: Domain-specific training datasets $D_1, \ldots, D_T$, experts $\mathcal{E} = \{e_1, \ldots, e_T\}$

Initialize empty map $\mathcal{P} = \{\}$ for the pilot embedding library
for each dataset D_i in $\{D_1, \ldots, D_T\}$ **do**
  Initialize an empty list $\mathcal{L}_i = [\,]$
  for each instance x_j in D_i **do**
    $e_{\text{max}} = \arg\max_{e_i \in \mathcal{E}} \text{Performance}(e_i, x_j)$
    Add pair (x_j, e_{max}) to $\mathcal{L}_i$
  end for
  for each expert e_m in $\mathcal{E}$ **do**
    $Group_m = \{x_j \mid e_{\text{max}} = e_m \text{ for } (x_j, e_{\text{max}}) \text{ in } \mathcal{L}_i\}$
    if $Group_m$ is not empty **then**
      $\mathbf{c}_m = \text{Centroid}(\text{BaseEncoder}(Group_m))$
      $\mathcal{P}[e_m].\text{append}(\mathbf{c}_m)$
    end if
  end for
end for
Output: Pilot embeddings library $\mathcal{P}$

lected expert alongside the base encoder.

During training, we fix the base retrieval model and only finetune the specialized experts, one for each target domain using domain-specific training data. We use Contriever (Izacard et al. 2021) as our base encoder, and our experts are parameter-efficient LoRA (Hu et al. 2021) modules. We also pre-compute for each domain a set of representative *pilot embeddings* that will help us route queries to appropriate experts. We refer to the mappings between pilot embeddings to the associated trained expert for each domain as the *pilot embedding library*. This overall process is only performed once. During inference, when given an input query, a *routing mechanism* determines the appropriate expert by calculating the similarity score between the input query embedding and the pilot embeddings in the pilot embedding library, and then choosing the expert with the highest average similarity score. This design allows for the flexible addition or removal of domain-specific experts without requiring any further training of the routing mechanism. To get into specifics:

Experts For each domain D_i, where $i = 1, \ldots, T$ and T is the total number of domains, we train an expert LoRA module e_i using the corresponding domain dataset. After the training step, we have a total of T different experts, $\mathcal{E} = \{e_1, e_2, \ldots, e_T\}$, with each expert e_i specialized for a specific domain.

Pilot Embedding Library To construct the pilot embedding library, given a domain-specific training dataset $D_i = \{x_1, \ldots, x_n\}$ where x_j is an instance in D_i, we perform inference using all experts $\mathcal{E}$ to identify which expert provides the most suitable representative embedding for each instance as shown in Algorithm 1. For each instance x_j, we select e_{max}, the expert that demonstrates the highest performance, defined as $e_{\text{max}} = \arg\max_{e_i \in \mathcal{E}} \text{Performance}(e_i, x_j)$. This process produces pairs (x_j, e_{max}) for all instances in the dataset D_i. The intuition here is that e_{max} for x_j doesn't have to be the expert trained on the source D_i that contains x_j.

Next, with the constructed pairs (x_j, e_{max}), we group them by the ones that have the same e_{max}. This results in T groups, one for each domain ($Group_m, m = 1, \cdots, T$), where each $Group_m$ contains list of instances x_j all sharing the same e_{max}. If the $Group_m$ is not empty, we extract all embeddings for instances in the group using the base encoder (BaseModel), and calculate the average, or centroid, embedding $\mathbf{c}_m$, which is taken as the pilot embedding for the domain[1]. 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 expert, representing the most suitable one for that domain. When $Group_m$ is empty, we skip this step, so the number of pilot embeddings for D_i could be less than T.

By repeating this process across all domain-specific training datasets $D_1, \ldots, D_T$, we obtain a maximum of T^2 pilot embeddings: T domain-specific training datasets times T groups per dataset.

Routing Mechanism 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 expert, resulting in a mean similarity score for each expert. The expert 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 a single base encoder model trained on the same dataset in a Multi-Task manner and a single base encoder model trained on a large-scale general-domain dataset MSMARCO. Following previous works (Muqeeth et al. 2024; Jang et al. 2023), we also evaluate two oracle settings: DatasetOracle and InstanceOracle[2]. The DatasetOracle 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 InstanceOracle 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 expert to calculate the probability of that expert being selected. The expert with the high-

[1]We also experiment with k-means clustering of different numbers of k, the number of centroid embeddings, and having a single centroid embedding (k=1) yields the highest performance, as additional centroids often act as distractors. Details are in the supplementary materials.

[2]DatasetOracle and InstanceOracle correspond to Best Individual and Oracle, respectively, in prior works Jang et al. (2023) and Muqeeth et al. (2024)

est 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 expert 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 baselines and training methods are in supplementary materials.

Dataset We use the provided training[3] and test sets in the BEIR benchmark (Thakur et al. 2021). We inspect BEIR domain splits using embeddings from our base encoder, a pre-trained Contriever model, in Figure 2. We observe that datasets like MSMARCO (Campos et al. 2016) and ArguAna (Wachsmuth, Syed, and Stein 2018) tend to have widely dispersed embeddings, indicative of their "general-domain" nature, while other datasets like HotpotQA (Yang et al. 2018), NFCorpus (Boteva et al. 2016), SciFact (Wadden et al. 2020), and FiQA (Maia et al. 2018) tend to have compact and tightly clustered instances, indicative of their "domain-specific" nature. Some like Quora (Iyer, Dandekar, and Csernai 2017) have high dispersion in their queries but have tightly clustered contexts. Where needed, we may use acronyms for datasets: ArguAna (AR), Quora (QU), MSMARCO (MS), HotpotQA (HO), SciFact (SF), NFCorpus (NF), FiQA (FI), SciDocs (SD), and TREC-COVID (TR).

Hyperparameters We use the pre-trained Contriever (Izacard et al. 2021) as our base encoder and train experts (LoRA) according to the settings in Lee et al. (2023), with a rank of 8, an alpha of 32 per expert, thereby training approximately 0.5% of the parameters (about 1M parameters) per expert. 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. For brevity, we focus on presenting results for which experts are applied only to the query encoder, keeping the context encoder frozen. We include results of applying experts to the context encoder in the supplementary materials.

Results

Overall Performance

Table 1 shows the performance of ROUTERRETRIEVER using seven domain-specific experts compared to baseline models, evaluated on test sets of corresponding experts. **ROUTERRETRIEVER outperforms both single model baselines**—Multi-Task training over the same training data mix as well as training only on MSMARCO—even without an MSMARCO expert.[4] And including an MSMARCO expert further improves performance for most domains. These results underscore the importance of having separate embedding models (experts) for each domain and dynamically selecting the most appropriate expert for each query rather than relying on a single model to handle multiple domains. We include additional results with different combinations of experts in the supplementary material.

Figure 2: TSNE visualization of contriever embeddings for queries when sampled 100 instances from each dataset. We see high dispersion "general-domain" datasets like ArguAna and MSMARCO (blue) while "domain-specific" datasets like HotPotQA (green), NFCorpus (grey), SciFact (pink), and FiQA (purple) are tightly clustered.

Comparing Different Routing Techniques

We experiment with different routing techniques commonly used in language modeling and compare them to our proposed routing mechanism. Results in Table 1 show that **the routing technique used in ROUTERRETRIEVER consistently achieves the highest performance**. In fact, ClassificationHeadRouter and ExpertClassifierRouter approaches tend to underperform compared to simply using a single retriever trained solely on MSMARCO. DatasetRouter, which is the closest to ROUTERRETRIEVER, tends to show higher performance than the single model trained on MSMARCO but also often still shows lower performance than ROUTERRETRIEVER. These results suggest that **routing techniques developed for language modeling may not generalize well to information retrieval**. We hypothesize that the differences in the effectiveness of routing techniques between language modeling and information retrieval can be explained

[3] In our early experiments, we noted that some training datasets in BEIR were so small that domain-specific models were underperforming MSMARCO on those target domains simply due to lack of training data. To conduct a proper study of routing over experts, we had to first ensure that the respective experts were reasonably well-trained. As such, for our experiments we also use the generated queries provided by BEIR at https://huggingface.co/BeIR.

[4] For fair comparison, we ensure the number of training instances of used ROUTERRETRIEVER and Multi-Task mix does not exceeds the number of training instances in MSMARCO.

		MSMARCO	Quora	Arguana	HotpotQA	NFCorpus	SciFact	FiQA	Avg
Retrievers	Single model on MSMARCO	**25.7**	**84.1**	37.2	57.6	31.7	67.2	28.8	47.5
	Single model with Multi-Task	22.4	82.0	36.9	52.1	32.9	69.4	28.9	46.4
	ROUTERRETRIEVER (w/o MSMARCO expert)	22.2	83.6	**39.5**	**59.5**	**33.4**	**76.0**	**30.5**	**49.3**
Routing	ExpertClassifierRouter	**23.8**	82.5	37.9	53.1	31.5	67.1	29.1	46.4
	ClassificationHeadRouter	22.6	83.4	38.5	52.8	32.7	69.6	28.2	46.8
	DatasetRouter	23.6	**83.9**	37.3	58.4	33.1	73.4	29.9	48.5
	ROUTERRETRIEVER (w/ MSMARCO expert)	23.0	83.8	**38.6**	**59.9**	**33.4**	**77.6**	**30.8**	**49.6**
Oracles	DatasetOracle	25.7	84.5	40.2	59.9	34.4	79.8	32.2	50.9
	InstanceOracle	34.5	89.9	48.5	66.6	39.0	85.4	39.6	57.6

Table 1: **Retrievers:** When trained on the same dataset size, ROUTERRETRIEVER consistently outperforms single model baselines (MSMARCO and Multi-Task) in terms of nDCG@10 on BEIR benchmark. **Routing:** ROUTERRETRIEVER also surpasses various standard routing techniques commonly used in language modeling. **Oracles:** ROUTERRETRIEVER achieves performance comparable to the DatasetOracle model. InstanceOracle indicates room for future work in router improvements.

	w/ Experts	w/o Experts
Single model on MSMARCO	47.5	31.6
Single model with Multi-Task	46.4	31.2
ROUTERRETRIEVER (w/ MSMARCO expert)	**49.6**	**31.9**
DatasetOracle	50.9	34.2
InstanceOracle	57.6	41.5

Table 2: ROUTERRETRIEVER not only shows high performance (nDCG@10) for datasets with trained experts but it also generalizes to those without experts. "w/ Experts" results are taken from Table 1. "w/o Experts" averages results on seven other BEIR test sets that lack training sets.

from the following perspective: 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 care to ensure precise routing. We achieve this by designing a routing mechanism around embedding similarities, which leans into the strengths of our encoder models. We are excited to see more sophisticated routing methods inspired by retrieval-specific designs in future works, especially closing the gap to InstanceOracle performance.

Zero-shot Generalization to Unseen Domains

We've demonstrated ROUTERRETRIEVER's effective use of domain-specific training data, but what of unseen test sets that don't have corresponding trained experts? In Table 2, we evaluate our models for true zero-shot generalization to seven more BEIR test sets: Touche-2020 (Thakur et al. 2024), Climate-FEVER (Diggelmann et al. 2020), DBPedia (Hasibi et al. 2017), NaturalQuestions (Kwiatkowski et al. 2019), FEVER (Thorne et al. 2018), TREC-COVID (Roberts et al. 2020), and SciDocs (Cohan et al. 2020). We find **the benefits of ROUTERRETRIEVER extend beyond the datasets for which specific experts were trained**. We provide dataset-specific results in supplementary materials.

Training and Inference Efficiency

ROUTERRETRIEVER achieves high training efficiency by using parameter-efficient LoRA experts, which account for only 0.5% of the parameters per expert. This makes the addition of new experts insignificant in terms of total parameter count. It uses the same amount of training data as any 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: first to identify the appropriate expert (routing), and second to generate the final query embedding. Improving the computation efficiency of this routing technique is a direction for future work. Detailed analysis over the efficiency is in supplementary details.

Analysis

Impact of Dataset Size when Training Experts

Figure 3 shows the relationship between amount of training data and performance of a single expert retriever. For in-domain evaluation datasets, performance generally improves as the number of training instances increases. However, in out-of-domain evaluation datasets, simply increasing the number of training samples does not necessarily lead to better performance. Interestingly, when testing out-of-domain, 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 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, broad coverage and diversity of the training dataset has a more significant impact on out-of-domain performance.

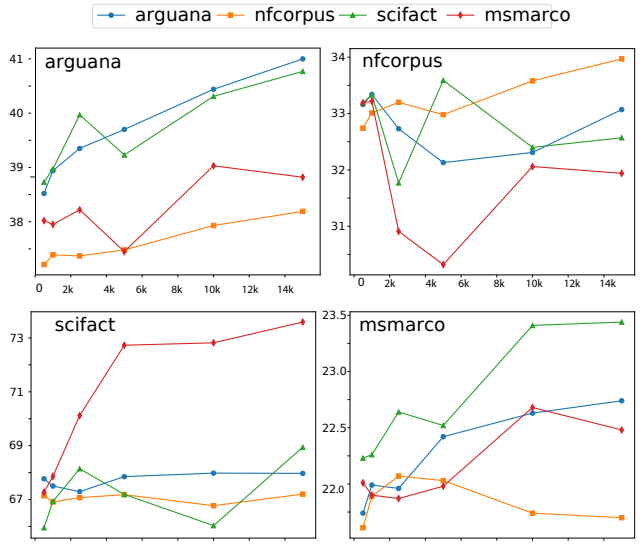


Figure 3: Single expert performance (nDCG@10; y-axis) against number of training instances (x-axis). Each line color represents the training dataset used, and each plot is a BEIR test dataset. As we increase training set size, in-domain performance increases rapidly, but may not transfer to improved out-of-domain performance.

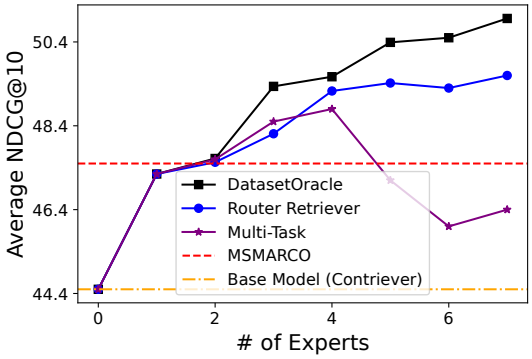


Figure 4: Average nDCG@10 (y-axis) by the number of experts (x-axis) for various models. ROUTERRETRIEVER tends to show improved performance as number of experts increases, outperforming a single MSMARCO-trained model even with just three experts despite less training data.

Impact of Number of Experts

Figure 4 shows the relationship between number of experts and performance of ROUTERRETRIEVER. It outperforms the single MSMARCO-trained model even with just three experts, 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. The performance of multi-task training tends to fluctuate as the number of domains (experts) increases. We hypothesize that with a large number of domains, the model struggles to find the optimal embedding for general cases due to high variance across training datasets.

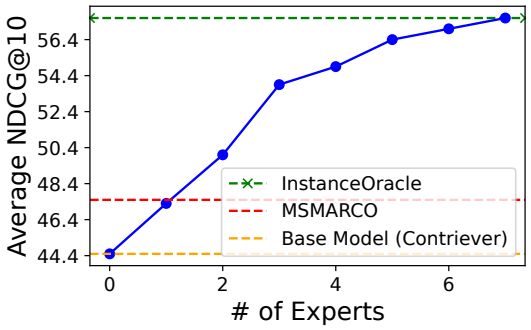


Figure 5: Average instance-level oracle routing performance nDCG@10 (y-axis) by the number of available experts (x-axis). The improvement rate tends to be high when adding experts initially followed by diminishing returns.

Yet, Figure 4 also shows diminishing returns in ROUTERRETRIEVER as we increase number of experts, but consistent increase in DatasetOracle. To further study whether this is due to need for better routing, we experiment with InstanceOracle but varying the pool of available experts. Figure 5 shows as the number of experts increases, InstanceOracle performance also improves quickly before encountering some diminishing returns. In repeating these experiments, we found this is true regardless of the expert combinations or order in which they're added. Overall, we interpret the results to mean ROUTERRETRIEVER's routing technique tends to be more distracted as more experts are added, which could motivate future work on scaling router fidelity closer to InstanceOracle to handle the higher complexity of more experts. Details are in the supplementary materials.

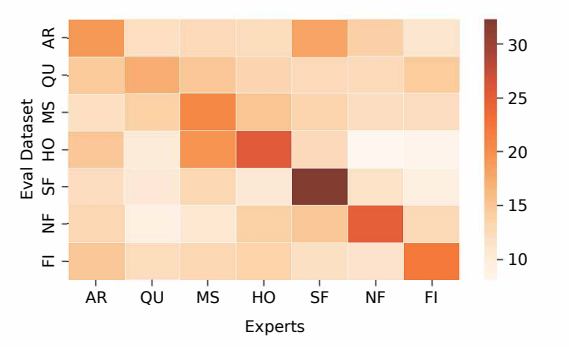
Table 3 shows results from experiments in which we sequentially add experts to ROUTERRETRIEVER. At low expert counts, each addition of a new expert can dramatically change performance across tasks that previously had an in-domain expert. For example, adding a HotpotQA expert caused performance on ArguAna to drop from 40.1 to 38.5 and SciFact from 76.7 to 72.2, while causing the expected improvement in HotpotQA to increase from 55.3 to 59.2. At higher expert counts, these side-effects are much more muted. For example, adding SciDocs or TREC-COVID experts to a seven-expert ROUTERRETRIEVER that improves performance for SciDocs (14.8 to 16.3) and TREC-COVID (44.9 to 56.2), but doesn't change overall performance for the other categories. Detailed results are provided in the supplementary materials.

Analyzing Routing Errors using InstanceOracle

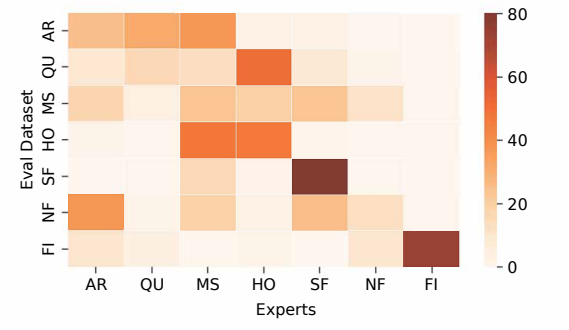
In Figure 6, we hope to understand the gap in performance between ROUTERRETRIEVER routing and InstanceOracle routing by inspecting which instances are routed to which experts by either method. First, Figure 6a shows that general-domain datasets like ArguAna and MSMARCO yield experts that can handily tackle instances from other di-

Start	Addition	AR	NF	SF	FI	HO	QU	MS	SD	TR
Unstable additions with few experts										
AR, NF, SF, FI	-	40.1	32.3	76.7	30.7	55.3	83.2	22.1	15.1	43.1
AR, NF, SF, FI	+ HotpotQA	38.5	33.0	72.2	27.9	59.2	82.7	22.3	15.9	43.6
AR, NF, SF, FI, HO	+ Quora	39.5	33.4	76.0	30.5	59.5	83.6	22.2	15.1	44.3
Stable additions with more experts										
AR, NF, SF, FI, HO, QU	+ MSMARCO	38.6	33.4	77.6	30.8	59.9	83.8	23.0	14.8	44.9
AR, NF, SF, FI, HO, QU, MS	+ SciDocs	38.8	32.7	76.9	30.3	59.8	84.1	22.9	16.3	44.7
AR, NF, SF, FI, HO, QU, MS	+ TREC-COVID	38.4	32.7	77.3	31.4	59.9	83.9	22.7	14.6	56.2

Table 3: Performance (nDCG10) across BEIR datasets while sequentially adding more experts to ROUTERRETRIEVER.



(a) InstanceOracle



(b) ROUTERRETRIEVER

Figure 6: For each evaluation dataset (y-axis), how often is each expert chosen (x-axis)? Darker cells mean more frequent selection. Diagonal entries mean in-domain selection. **(a)** While "general" experts like AR and MS appear well-suited to tackle instances from other datasets (darker columns), instances from certain datasets like SF and NF must be routed to the in-domain expert (sparse columns with single dark concentration). **(b)** ROUTERRETRIEVER has sparser routing behavior, tending towards following dataset boundaries, which explains similar results as DatasetOracle.

verse datasets; hence, they receive an even distribution of queries from InstanceOracle. However, for domain-specific datasets like SciFact or NFCorpus, the best performance is typically achieved by the expert trained specifically on that domain. Figure 6b then shows that ROUTERRETRIEVER's expert selection tends to be much sparser, prioritizing routing instances to their source dataset's expert. This explains the similar performance seen in Table 1 between ROUTERRETRIEVER and DatasetOracle, and motivates future work on more powerful routing mechanisms. We add a detailed error analysis of our routing technique in the supplementary materials.

Conclusion

In this paper, we introduce ROUTERRETRIEVER, a retrieval model that leverages a mixture of domain-specific expert embeddings, guided by a routing mechanism to select 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. Our results highlight the crucial role of domain-specific experts in improving retrieval performance across diverse domains. Yet, there remains much more room for improvement, as indicated by the higher performance of an instance-level oracle router compared to our method. We hope our work spurs the broader research community to search for more powerful methods for training expert retrievers and combining them with efficient routing techniques.

Acknowledgments

We thank Nandan Thakur, Orion Weller, Jiyeon Kim, Hanseok Oh, and the Semantic Scholar Research team at Ai2 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.

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.

Iyer, S.; Dandekar, N.; and Csernai, K. 2017. First Quora Dataset Release: Question Pairs.

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, K.; Lo, K.; Gonzalez, J. E.; and Klein, D. 2023a. Decomposing Complex Queries for Tip-of-the-tongue Retrieval. In *Conference on Empirical Methods in Natural Language Processing*.

Lin, S.-C.; Asai, A.; Li, M.; Oğuz, B.; Lin, J. J.; Mehdad, Y.; tau Yih, W.; and Chen, X. 2023b. 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*.

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.; Bonifacio, L.; Fröbe, M.; Bondarenko, A.; Kamalloo, E.; Potthast, M.; Hagen, M.; and Lin, J. 2024. Systematic Evaluation of Neural Retrieval Models on the Touché 2020 Argument Retrieval Subset of BEIR.

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*.