Unsupervised Query Routing for Retrieval Augmented Generation

Anonymous ACL submission

Abstract

Query routing for retrieval-augmented generation aims to assign an input query to the most suitable search engine. Existing works rely heavily on supervised datasets that require extensive manual annotation, resulting in high costs and limited scalability, as well as poor generalization to out-of-distribution scenarios. To address these challenges, we introduce a novel unsupervised method that constructs the "upper-bound" response to evaluate the quality of retrieval-augmented responses. This evaluation enables the decision of the most suitable search engine for a given query. By eliminating manual annotations, our approach can automatically process large-scale real user queries and create training data. We conduct extensive experiments across five datasets, demonstrating that our method significantly enhances scalability and generalization capabilities.

1 Introduction

Retrieval Augmented Generation (RAG) (Lewis et al., 2020; Gao et al., 2023) typically begins with a retrieval phase where user queries are scoured through search engines like Google and Bing to gather relevant background documents before engaging large language models (LLMs) (OpenAI, 2023; Qwen Team, 2023). In today's digital landscape, even among major comprehensive search engines, each has its strengths and weaknesses, offering unique features and capabilities. Consequently, *Query Routing* (Sugiura and Etzioni, 2000; Wang et al., 2024; Mu et al., 2024) seeks to direct queries to the most suitable search engines to optimize performance and reduce costs.

Current query routing methods (Shnitzer et al., 2023; Mu et al., 2024) primarily use open-sourced *(query, answer)* paired data to create training labels. Specifically, by leveraging these gold-standard answers, they assess the quality of retrieval-augmented responses for queries routed through different search engines, thereby identifying the most effective engine for each query. However, the reliance on annotated paired data presents challenges such as limited diversity and high annotation costs, which significantly restrict scalability. Moreover, there exists a pronounced discrepancy between the distribution of *public datasets* and *real user queries*, which severely impacts the out-of-distribution generalization (Shnitzer et al., 2023) of these methods, limiting their effectiveness in real-world applications. To enable models to generalize effectively to actual user queries, it is essential to train them directly on real user queries. Nevertheless, real user queries often lack gold-standard answers, presenting a major challenge in determining the appropriate search engine assignment, particularly when *no definitive answer is available*.

Search Strategies	Qwen2-max	GPT4
No-RAG	2.189	2.014
Using one search engine		
Quark	2.651	2.456
Bing	2.615	2.390
Google	2.610	2.420
Using two search engines		
Quark+Bing	2.689	2.495
Quark+Google	2.701	2.486
Bing+Google	2.674	2.478
Using three search engines		
Quark+Bing+Google	**2.727**	**2.521**

Table 1: RAG results across various search strategies. We calculate average scores on all examples. No-RAG denotes that LLMs respond to the query without retrieval. Scores with **bold** denote the best results.

To address this challenge, we must transform our problem into a labelling-free form. Since query routing fundamentally involves identifying the optimal assignment within the limited computing resources, such as inference time and budget, we can envision an ideal situation in which we pos-

sess infinite computing resources. In this case, an intuitive and obvious strategy is to search a query across all search engines, then merge multi-sourced documents for RAG[1]. Due to the complementarity of different search engines (Mu et al., 2024), the merged multi-sourced documents should have a better recall than the single-sourced documents. That is, multi-sourced responses are likely to exhibit higher quality than single-source responses. To support this assertion, we calculate the RAG results across various search strategies and different LLMs, as illustrated in Table 1. We find that utilizing more search engines consistently leads to improved results. This trend is observed across multiple LLMs. Based on this prior knowledge, we can view the multi-sourced responses as an upper bound for unsupervised query routing. Now, let us return to the practical situation of limited computing resources. To evaluate the quality of a single-sourced response, we can compare it to the corresponding multi-sourced response, which serves as an upper bound. By assessing the similarity between the single-sourced response and its upper bound, we are able to determine its quality. If the single-sourced response shows a high level of similarity to its upper bound, this suggests that the query should be assigned to the corresponding search engine. This chain of thought motivates us to develop an unsupervised query routing method that directly leverages real user data to generate training labels in an automated manner, eliminating the need for manual annotation.

In this work, we propose a novel unsupervised query routing method that directly creates training data without human annotations. Our method mainly consists of four steps. In the first step, we search every user query in only a single search engine to build single-sourced responses. In the second step, we search every user query across all available search engines to establish multi-sourced responses, which are used as the upper bound for unsupervised query routing. Next, given a query and its corresponding single-sourced responses, we use several automated metrics to evaluate the quality of the single-sourced responses, allowing us to determine which search engine best fits the query. Finally, by constructing training data automatically, we train our routing model. Our method operates without the need for manual annotation, enabling the automation of large-scale processing of real-world user queries and facilitating the construction of training data. Consequently, our approach can be effectively scaled and has a strong generalization capability for real user queries.

In summary, our contributions are as follows. (1) We introduce a novel unsupervised query routing method capable of automatically processing large-scale, unlabeled real user data for model training. This simple yet effective approach significantly reduces annotation costs and offers excellent scalability potential. (2) As far as we know, we are the first to explore unsupervised query routing within the RAG scenario. Our method serves as a universal framework, holding significant promise for providing valuable insights into the field of tool learning. (3) We validate our method across five datasets in a non-i.i.d. setting. The experiments demonstrate that our approach exhibits **i)** outstanding generalization ability, **ii)** excellent scalability, and **iii)** consistent effectiveness across multiple LLMs.

2 Related Work

Query Routing for LLMs Nearly monthly released LLMs (Qwen Team, 2023; Dubey et al., 2024) are trained with various data, resulting in a variety of strengths and weaknesses in versatile downstream tasks (Jiang et al., 2023). Therefore, researchers (Narayanan Hari and Thomson, 2023; Lu et al., 2023; Liu et al., 2024; Shnitzer et al., 2023; Chen et al., 2023b; Šakota et al., 2024) are devoted to learning the optimal assignment of queries to different LLMs. The basic idea is to leverage the complementarity of various LLMs to achieve superior performance compared to any single model across a range of tasks. Notably, these methods typically require substantial amounts of annotated training data, leading to high costs for label collection. In contrast to these approaches, we explore query routing across different search engines and propose an unsupervised method, thereby eliminating the need for manual annotation and significantly reducing annotation costs.

Query Routing for Search Engines Early research focused on topic-specific search engines (Manber and Bigot, 1997; Sugiura and Etzioni, 2000), which often suffer from limited coverage. Alternatively, other conventional systems (Gravano

[1]For simplicity, we denote the generated response when retrieving from multiple tools as the multi-sourced response. Correspondingly, we denote the generated response when retrieving from only one tool as the single-sourced response.

et al., 1999; Liu, 1999) target on general-purpose search engines, but they require access to the complete internal databases associated with each engine. To address this issue, (Mu et al., 2024) recently utilized annotated (query, answer) data to train neural models that can find the most appropriate general-purpose search engine for a given query. However, these supervised methods rely heavily on annotated data. In contrast, our method does not require manual annotation and directly uses real user queries for training, thus offering promising scalability and generalization capabilities.

3 Preliminaries

Given a user query q and M search tools $\{T_m\}_{m=1}^{M}$, the core task of query routing is to create annotated training data in the form (q, π), where π represents the search engine index indicating which search engine should be selected to address the query.

3.1 Briefing of Previous Supervised Methods

Existing supervised methods (Shnitzer et al., 2023; Mu et al., 2024) rely heavily on annotated (query, answer) paired data. Specifically, using each search tool T_m, they are able to obtain the retrieval-augmented response r_m, respectively. By calculating the similarity between r_m and the gold answer r_{gold}, they can obtain the labeled index:

$$\pi = \arg\max_{m} \text{similarity}(r_m, r_{\text{gold}}). \tag{1}$$

However, as mentioned earlier, public datasets tend to be relatively small in scale, lack diversity, and significantly differ from actual user queries, which restricts their scalability and generalization ability.

3.2 Problem Formulation

We devise an unsupervised method that relies solely on user queries and search tools to generate training data. Formally, we first obtain the single-sourced responses using every search tool T_m:

$$\begin{aligned} \mathbf{DOC}_m &= T_m(q) \\ r_m &= \text{LLM}(q, \mathbf{DOC}_m), \end{aligned} \tag{2}$$

where $\mathbf{DOC}_m$ denotes the single-sourced documents retrieved from T_m. Then, by aggregating retrieved documents from multiple search tools, we obtain the upper-bound multi-sourced response:

$$\begin{aligned} \mathbf{DOC}^\star &= T_1(q)||\cdots||T_M(q) \\ r^\star &= \text{LLMs}(q, \mathbf{DOC}^\star), \end{aligned} \tag{3}$$

where $\cdot||\cdot$ denotes the merging operation, $\mathbf{DOC}^\star$ denotes the multi-sourced documents, and $r^\star$ denotes the upper-bound multi-sourced response. Finally, by using the upper-bound $r^\star$ as the anchor, we evaluate the quality of every single-sourced response, which allows us to obtain the training label for q:

$$\pi = \text{Metrics}(q, \text{DOC}^\star, r_m, r^\star), \tag{4}$$

where the details of used metrics are shown in § 4.3. By comparing Equation 1 with Equation 4, it can be found that through a series of transformations, we use upper-bound multi-sourced $r^\star$ instead of the manually annotated r_{gold}, thus achieving the goal of unsupervised query routing.

4 Methodology

The overall framework of our method is shown in Figure 1. Our method mainly consists of four steps: (1) **Single-sourced Response Generation** which generates single-sourced responses for each search tool; (2) **Upper-bound Construction** which constructs the multi-sourced responses; (3) **Label Construction** which utilizes the single-sourced response and its corresponding multi-sourced upper-bound to create training labels; and (4) **Model Training** where we train our routing model.

4.1 Single-sourced Response Generation

As stated in Equation 2, for each tool T_m, we search q within T_m to retrieve $\mathbf{DOC}_m$ which consists of k documents. Next, we compile the query q and the retrieved document $\mathbf{DOC}_m$ into a prompt, which is then fed into a LLM to generate the single-sourced response r_m. The prompt is shown in Figure 5. For the LLM, we employ Qwen2-max[2] (Yang et al., 2024) to generate responses. This process is repeated for each search tool T_m, resulting in a collection of single-sourced responses $\{r_m\}_{m=1}^{M}$.

4.2 Upper-bound Construction

Given q and $\{T_m\}_{m=1}^{M}$, we can obtain $\mathbf{DOC}_m$ for each T_m (Equation 2), respectively. The question is how to retain k documents from $k * M$ documents to achieve a fair comparison to § 4.1. Previous works generally utilize the text reranker (Li et al., 2023; Chen et al., 2023a) to retain top-ranked documents. However, these rerankers require extra training and may not generalize well to black-box search engines. Therefore, we opt to forgo the re-ranking process and instead utilize the inherent sorting algorithm of the search engine directly. Specifically, for $\mathbf{DOC}_m$ that contains the ordered k documents, we only use the top-ranked k/m

[2]https://dashscope.aliyuncs.com/compatible-mode/v1

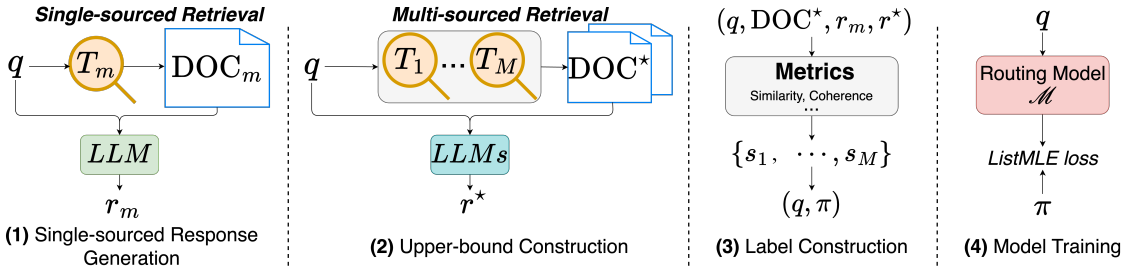


Figure 1: The overall framework of our method, which mainly consists of four steps. We automatically construct supervision information for training our routing model.

documents. That is, for $\{\mathbf{DOC}_m\}_{m=1}^{M}$, we finally reserve the multi-sourced documents $\mathbf{DOC}^\star$ which contains k documents. Then, we pass q and $\mathbf{DOC}^\star$ to LLMs to generate the multi-sourced response $r^\star$ (Equation 3). Notably, we consider multiple LLMs, including Qwen2-max and GPT4, for generating $r^\star$ to improve the robustness of $r^\star$. This is motivated by the previous works of text generation (Bahdanau et al., 2014; Sutskever et al., 2014) in which the gold references usually contain multiple texts.

4.3 Label Construction

We devise two metrics, i.e., similarity and coherence scores, to assess the quality of $\{r_m\}_{m=1}^{M}$. Considering multiple scores for each r_m, we adopt a normalization schema to get the overall score, so that obtain the training labels.

Similarity Score We first use BertScore (Zhang et al., 2019) to evaluate the similarity between r_m and its upper-bound $r^\star$:

$$s_m^{\text{Bert}} = \text{BertScore}(r_m, r^\star). \tag{5}$$

The intuition is that the higher the value of s_m^{Bert}, the better the quality of r_m.

Coherence Score Besides Similarity, we also care about the coherence between the query and the single-sourced response, i.e., whether r_m can effectively solve q. Therefore, we use a LLM, e.g., Qwen2-max, to determine which one better solves q in a pair of single-sourced responses. Without loss of generality, we define the paired single-sourced responses as r_a and r_b. Taken q, the multi-sourced documents $\mathbf{DOC}^\star$, r_a, and r_b as inputs, the LLM determines the one with better quality[3]:

$$r_a \quad \text{or} \quad r_b \longleftarrow \text{LLM}(q, \mathbf{DOC}^\star, r_a, r_b). \tag{6}$$

Through $\frac{M*(M-1)}{2}$ comparisons, we can obtain the overall ranking relationship (in ascending order) among $\{r_m\}_{m=1}^{M}$[4]. We define the coherence score s_m^{Coh} as the index of r_m in the overall ranking. The larger the index, the higher the score.

Overall Score Having obtained s_m^{Bert} and s_m^{Coh} of r_m, we devise a normalization schema to get the overall score of r_m, which helps to mitigate the impact of varying dimensions of multiple metrics. For example, we calculate the mean μ^{Bert} and standard deviation σ^{Bert} of the similarity score across all examples. Then, the normalized similarity score and coherence score are calculated as follows:

$$\overline{s_m^{\text{Bert}}} = \frac{s_m^{\text{Bert}} - \mu^{\text{Bert}}}{\sigma^{\text{Bert}}}, \quad \overline{s_m^{\text{Coh}}} = \frac{s_m^{\text{Coh}} - \mu^{\text{Coh}}}{\sigma^{\text{Coh}}}. \tag{7}$$

The overall score of r_m is calculated as:

$$s_m = \frac{\overline{s_m^{\text{Bert}}} + \overline{s_m^{\text{Coh}}}}{2}. \tag{8}$$

Converting to Training Labels After obtaining $\{s_m\}_{m=1}^{M}$, we sort them in ascending order. We use the indices of $\{s_m\}_{m=1}^{M}$ in the ordered sequence as the label, denoted as $\pi \in \mathcal{R}^M$. π actually represents a permutation of $\{1, 2, \cdots, M\}$, indicating the priority of routing q to $T_1, \cdots, T_M$, respectively. This kind of label enables us to create a listwise ranking loss to effectively leverage the priority relationships between each pair of tools.

4.4 Model Training and Inference

Given the labeled (q, π) examples, we train a neural model $\mathcal{M}$ to predict the scores of routing q to $\{T_m\}_{m=1}^{M}$, respectively:

$$\mathbf{p} = \mathcal{M}(q) \tag{9}$$

where $\mathbf{p} = \{p_1, \cdots, p_m, \cdots, p_M\}$, p_m denotes the score of routing q to T_m. Then, we adopt the listwise ranking loss ListMLE (Guiver and Snelson, 2009) to optimize model parameters:

$$\mathcal{L} = \text{ListMLE}(\mathbf{p}, \pi). \tag{10}$$

[3]The prompt is shown in Appendix A, Figure 6.

[4]We discard the illegal ranking output by LLMs (like "$A > B, B > C, C > A$").

Search Strategies	Qwen2-max					GPT4				
	WebQA	PrivateMH	NLPCC-MH	CDQA	SogouQA	WebQA	PrivateMH	NLPCC-MH	CDQA	SogouQA
No-RAG	4.609	2.77	1.803	2.543	3.31	3.927	1.924	2.647	2.538	3.715
Using one search engine										
Quark	**4.833**	3.38	2.258	3.691	4.484	4.321	2.657	2.735	3.73	4.42
Bing	4.79	3.309	2.151	3.661	4.437	4.168	2.461	2.737	3.559	4.33
Google	4.797	3.205	2.104	3.745	4.395	4.225	2.557	2.644	3.79	4.312
Using two search engines										
Quark+Bing	4.823	3.428	2.274	3.829	4.546	**4.323**	2.669	2.755	3.877	4.445
Quark+Google	4.822	3.418	2.261	3.917	4.487	4.285	2.682	2.701	3.925	4.412
Bing+Google	4.815	3.363	2.214	3.858	4.458	4.24	2.61	2.765	3.893	4.36
Using three search engines										
Quark+Bing+Google	4.808	**3.445**	**2.352**	**3.993**	**4.584**	4.285	**2.705**	**2.796**	**3.962**	**4.45**

Table 2: The Correctness results under different combinations of LLM and search tools. Scores with **bold** denote the best result. Values are calculated by averaging 3 random experiments.

During inference, our method only requires the input q to predict the scores of using each search tool, without actually calling any search tool.

5 Experiment

5.1 Experimental Settings

Datasets We use four public open-domain QA datasets, including NLPCC-MH (Wang and Zhang, 2018), CDQA (Xu et al., 2024), WebQA (Chen et al., 2019), SogouQA[5], and a private QA dataset, named PrivateQA, for evaluation. The statistics of the dataset and the details of the construction of PrivateQA are shown in Appendix B.

Search Tools and LLMs We consider 3 search tools, i.e., Quark, Bing, and Google, for document retrieval. For each query, we retrieve $k = 6$ documents. That is, in the upper-bound construction process, we only retain the top-ranked 2 documents for each search tool. We use Qwen2-max and GPT4 for response generation.

The Evaluation Metric Each test example is in the form of *(query, answer)*. For each query, we use the different combinations of LLMs and search tools to obtain the retrieval-augmented responses. We adopt the **Correctness** score implemented by Llama-index (Liu, 2022) as the metric to evaluate the quality of generated responses. For each query, we repeatedly generate three times and calculate the average correctness score. Since we have considered 3 search engines and 2 LLMs, we have 16 RAG strategies. The result is shown in Table 2. In the vast majority of strategies, using more search tools will lead to better results.

[5]https://aistudio.baidu.com/datasetdetail/17514

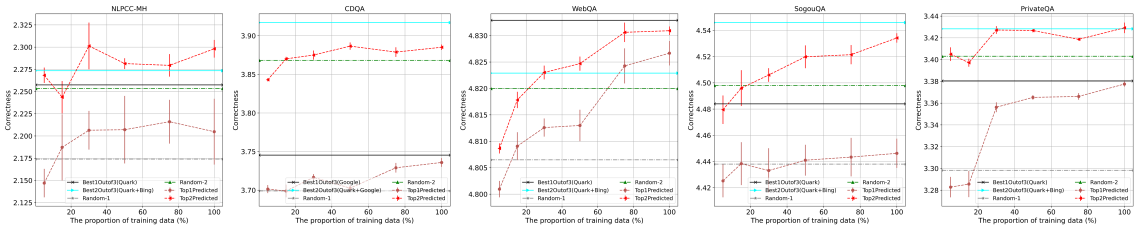
Data Source of Training Queries We utilize our in-house data to construct the training examples. After filtering out illegal samples (§ 4.3), we finally obtain around 110k training examples. It should be noted that there is a considerable distribution discrepancy between the training and test examples. As a result, the evaluation of our method can be regarded as in a non-i.i.d. setting.

Model Training Details We initialize our routing model as GTE-large (Li et al., 2023). Since our method is model-agnostic, we have also tried other backbone models, as shown in § 5.4. We utilize the AdamW optimizer with an initial learning rate of 5e-5 and a batch size of 32. The learning rate is linearly decayed to zero with a 10% warmup ratio over 10 training epochs. Our model is evaluated through 5 randomized experiments, and we report the average results. All experiments are conducted using the PyTorch toolkit on an Ubuntu server equipped with four V100 (32GB) GPUs.

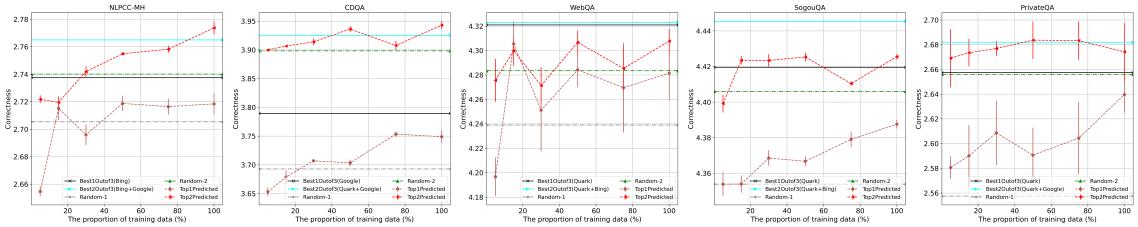
5.2 The Evaluation of Scalability

Our method, which eliminates the need for manual annotation, is designed to exhibit strong scalability. This means that as we increase the volume of training data, we are expected to witness a corresponding improvement in the performance of our model. In this section, we conduct a comprehensive evaluation of the scalability of our approach.

Setting We alter the proportions of training data to assess the scalability of our method. For each proportion of training data, we present the average result of five random experiments. We consider the following baselines for comparison: (1) **Random-1**



(a) The data scaling result when using Qwen2-max for response generation.



(b) The data scaling result when using GPT4 for response generation.

Figure 2: The data scaling result of our method. For **Best1Outof3** and **Best2Outof3**, we explicitly indicate the specific search strategy in parentheses. The values are obtained by averaging 5 random experiments.

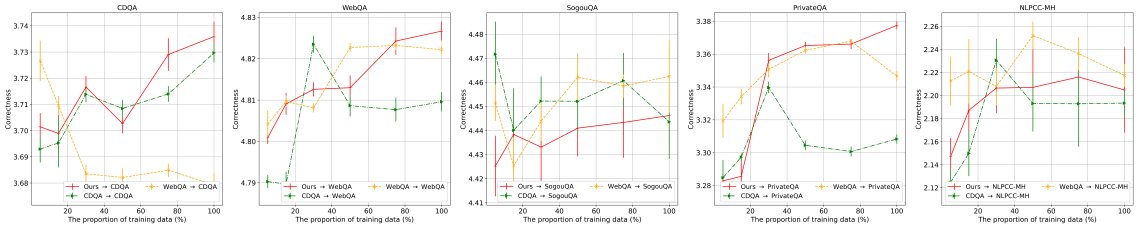


Figure 3: The evaluation result of the generalization abilities of different methods. We report the **Top1-Predicted** result. Values are calculated by averaging 5 randomized experiments. We try three different types of training examples on five test sets. "$A \rightarrow B$" means we train the model on A and evaluate the model on B. For example, "CDQA → WebQA" means we train the model on CDQA and evaluate the model on WebQA. Therefore, "CDQA → CDQA" and "WebQA → WebQA" are actually in an i.i.d. setting.

which randomly selects a tool from Quark[6], Bing, or Google for RAG; (2) **Random-2** which randomly selects a combination from Quark+Bing, Bing+Google, or Quark+Google for RAG; (3) **Best1Outof3** which is the most effective one among Quark, Bing, or Google; (4) **Best2Outof3** which is the most effective one among Quark+Bing, Bing+Google, or Quark+Google.

It should be noted that **Best1Outof3** and **Best2Outof3** may correspond to different specific retrieval strategies on different datasets. Correspondingly, for each query, our method will predict the score of each tool. Therefore, we consider **Top1-Predicted** and **Top2-Predicted**, where **Top1-Predicted** indicates that we choose the tool with the highest predicted score, and **Top2-Predicted** indicates that we choose the two tools with the highest predicted scores. Ultimately, we compare **Top1-Predicted** with **Best1Outof3**, and compare **Top2-Predicted** with **Best2Outof3**.

Result The result is shown in Figure 2. The random baselines show an unsatisfactory performance. In addition, we have the following observations.

- As the volume of training data increased, our method has demonstrated consistent and obvious performance improvements. Since our method eliminates the necessity of manual annotation, the sustained enhancement in performance highlights the effectiveness of leveraging a larger amount of real user queries, ultimately, it is promising to obtain more refined and satisfied outcomes in diverse applications.
- The effectiveness of our method is observed consistently across various LLMs and multiple datasets, underscoring the robustness and stabil-

[6]https://www.quark.cn/

ity of our approach. The key reason is that a wide variety of real user queries are directly utilized for the learning of query routing, which can effectively guarantee data diversity.

- On NLPCC-MH and CDQA datasets, our method outperforms the benchmarking baseline, demonstrating that it has learned the routing mechanism to some extent. This capability enables it to assign queries to the most suitable search engine, resulting in superior performance. While our method does not outperform the best baseline on some datasets, we attribute this limitation to the still insufficient volume of training data. Actually, query routing among general-purpose search engines is an extremely difficult task, since even humans struggle to determine which search engine should be selected to handle a query when they have no knowledge about black-box search engines. Fortunately, since we have observed the data scaling effect, we have reason to believe that if we continue to expand the data volume, our method will show even greater improvements and achieve optimal results.

Overall, by systematically varying the size of the training dataset, we observe compelling scalability for our method, which demonstrates its potential for utilizing more data for training.

5.3 The Evaluation of Generalization Ability

Our model theoretically has good generalization ability. The underlying principle is that by collecting sufficient real user queries, data diversity can be ensured, enabling the model to effectively generalize to unseen data. Consequently, in this section, we compare our approach with existing supervised methods to assess its generalization ability.

Setting Our method uses real user queries for training. As a comparison, we reproduce existing supervised methods (Shnitzer et al., 2023; Mu et al., 2024) that utilize annotated *(query, answer)* data for model training. More specifically, we use the training splits of CDQA and WebQA which contain 10k and 20k examples, respectively, to construct training examples (§ 3.1). Then, we use the obtained examples to train the routing models and evaluate them on the 5 test sets. By altering the types of training data, we observe the scaling effect of the learned models on the 5 test sets.

Result The result is shown in Figure 3. We have the following observations.

- In the independent and identically distributed (i.i.d.) settings, specifically in the "CDQA → CDQA" and "WebQA → WebQA" scenarios, supervised methods demonstrate a pronounced scaling trend. This indicates that augmenting the volume of training data correlates with sustained performance improvements, which aligns with human intuitive understanding.
- Conversely, in the other non-i.i.d. scenarios of supervised methods, the addition of more training data fails to yield similar performance gains. This suggests that traditional supervised methods struggle to generalize effectively in non-i.i.d. environments. Such findings highlight a critical limitation in existing approaches. That is annotated data is difficult to obtain, which greatly limits the diversity of data and thus restricts their generalization ability.
- Our method demonstrates a pronounced data scaling trend across all datasets, although the trend on NLPCC-MH is not as obvious as the trend on other test sets. This is primarily due to the fact that the data scale utilized by our approach significantly surpasses that of traditional supervised methods. Such an expansive data scale effectively enhances the diversity of the training data. This characteristic underscores the innovation of our approach, which is capable of automatically curating a wide array of diverse training data, eliminating the need for both time and cost consuming human annotation.

5.4 Ablation Study

Backbone	CDQA	WebQA	SogouQA	PrivateQA
Full (ours)	3.736	4.827	4.446	3.378
w/o Similarity	3.718	4.806	4.438	3.286
w/o Coherence	3.698	4.796	4.422	3.272

Table 3: The impact of different automatic metrics in the label construction step. Values are calculated by averaging 5 randomized trials.

The Impact of Different Evaluation Metrics We develop the following two ablated variants to validate the effectiveness of different metrics in the label construction step: (1) "w/ Similarity" means we only use the similarity score for label construction; (2) "w/ Coherence" means we only use the coherence score for experiments. The result is shown in Table 3. Both "w/ Similarity" and "w/ Coherence" show a performance drop. In addition, we have the following observations.

- The performance of "w/ Similarity" surpasses that of "w/ Coherence," indicating that the similarity metric is more effective. It is important to note that the similarity metric employs multi-sourced responses as the anchor, whose reliability has been validated by our experiments. This suggests that the similarity metric is a more objective measure, resulting in labels that are more robust. In contrast, the coherence metric relies solely on the knowledge within the LLM itself. However, due to inherent knowledge limitations in LLMs, it can generate incorrect information, leading to reduced effectiveness compared to the similarity metric. As evidence, we observe that LLMs may generate illegal ranking outputs, such as $A > B, B > C$, but $C > A$, in the coherence score calculation. These illegal outputs account for approximately 9% of the population.
- Although not as effective as the Similarity metric, the Coherence metric contributes to our method. The best result is achieved when combining two metrics. One possible explanation is that Similarity and Coherence assess different aspects of the generated single-sourced responses. Specifically, similarity measures how closely a single-sourced response aligns with its upper-bound, while Coherence evaluates the rationality of a single-sourced response in relation to the input query. As a result, these two metrics can effectively complement one another.

Backbone	CDQA	WebQA	SogouQA	PrivateQA
GTE-large (ours)	3.736	4.827	4.446	3.378
w/ Roberta-large	3.716	4.826	4.453	3.343
w/ Qwen-0.5B	3.727	4.808	4.419	3.290

Table 4: The result under different backbone models. Values are calculated by averaging 5 randomized trials.

The Result under Different Backbone Models We also investigate the impact of different backbone models. Besides GTE-large, we have also evaluated Roberta-large and Qwen2-0.5B, as they possess a comparable number of parameters to GTE-large. The result is shown in Table 4. We observe a minor result difference between GTE-large and Roberta-large. This is because they belong to the same type of models, and after training, they will all converge towards the distribution of the training data. However, Qwen2-0.5B shows a large performance drop compared to GTE-large and Roberta-large. We speculate that such decoder-only small models are less effective for regression tasks. Due to time and computational resource limitations, we are unable to try larger models, such as Qwen2-4 and Qwen2-7B, for experiments. Finally, we choose GTE-large as the backbone of our work.

Figure 4: The visualization of the high-frequency words. The words are translated from Chinese. We leave the original version in Appendix C.

5.5 Visualization

We conduct a further analysis of the constructed training labels. According to the label π, we categorize the queries into three distinct groups: those should be assigned to Quark, Bing, and Google, respectively. Then, we tally the high-frequency words presented in the queries of each group and visualize high-frequency words in each group, as depicted in Figure 4. In the "Bing" group, terms such as *"earthquake", "MacBook Pro", and "Hong Kong stock"* demonstrate a high frequency, indicating that Bing has good support for news hotspots as well as significant political and economic events. In the "Google" group, high-frequency words include "python, open-source, 5G", etc. This indicates that Google is better able to handle programming, technology, and other queries that require strong professionalism. This is quite intuitive, as Google, renowned as the world's leading search engine, offers the most extensive internal documentation available, enabling the platform to effectively tackle a diverse array of professional issues.

6 Conclusion

In this work, we present an unsupervised query routing approach. Given the insight that multi-sourced retrieval yields superior performance, we ingeniously design an automated process to assess the quality of single-sourced responses, thereby generating training labels without manual annotation. Experimental results demonstrate that our method has excellent scalability and generalization ability. We are confident that our approach will inspire new advancements within the community.

Limitations

Due to the high costs associated with large-scale data processing, we currently have access to only approximately 110k training samples. It is important to note that query routing among general-purpose search engines presents a particularly challenging problem, and it is likely that 110k examples are still insufficient for our needs. In the future, given a larger budget, we plan to conduct experiments using a greater number of examples. For our methodology, we have chosen smaller models instead of larger Qwen2-7B or Qwen2-14B as the backbone for two primary reasons: (1) Query routing demands low latency, whereas larger models tend to operate at slower speeds, which does not meet our requirements. (2) Tool selection necessitates that the model accurately predicts the score for each tool in relation to each query, which effectively constitutes a numerical prediction task. However, auto-regressive LLMs often exhibit significant challenges in numerical prediction.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2023a. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *Preprint*, arXiv:2309.07597.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023b. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*.

Yu Chen, Lingfei Wu, and Mohammed J. Zaki. 2019. Bidirectional attentive memory networks for question answering over knowledge bases. *ArXiv*, abs/1903.02188.

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

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Luis Gravano, Hector Garcia-Molina, and Anthony Tomasic. 1999. Gloss: text-source discovery over the internet. *ACM Transactions on Database Systems (TODS)*, 24(2):229–264.

John Guiver and Edward Snelson. 2009. Bayesian inference for plackett-luce ranking models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 377–384, New York, NY, USA. Association for Computing Machinery.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. *arXiv preprint arXiv:2306.02561*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

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

Jerry Liu. 2022. LlamaIndex.

Ling Liu. 1999. Query routing in large-scale digital library systems. In *Proceedings 15th International Conference on Data Engineering (Cat. No. 99CB36337)*, pages 154–163. IEEE.

Yueyue Liu, Hongyu Zhang, Yuantian Miao, Van-Hoang Le, and Zhiqiang Li. 2024. Optllm: Optimal assignment of queries to large language models. *arXiv preprint arXiv:2405.15130*.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2023. Routing to the expert: Efficient reward-guided ensemble of large language models. *arXiv preprint arXiv:2311.08692*.

Udi Manber and Peter Bigot. 1997. The search broker. In *USENIX Symposium on Internet Technologies and Systems (USITS 97)*.

Feiteng Mu, Yong Jiang, Liwen Zhang, Chu Liu, Wenjie Li, Pengjun Xie, and Fei Huang. 2024. Query routing for homogeneous tools: An instantiation in the rag scenario. *arXiv preprint arXiv:2406.12429v3*.

Surya Narayanan Hari and Matt Thomson. 2023. Tryage: Real-time, intelligent routing of user prompts to large language models. *arXiv e-prints*, pages arXiv–2308.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Alibaba Group Qwen Team. 2023. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Marija Šakota, Maxime Peyrard, and Robert West. 2024. Fly-swat or cannon? cost-effective language model choice via meta-modeling. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 606–615.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and

Current time is {timestamp}.
You are a helpful AI assistant. Your knowledge base contains the following content:

{Document 1}
...
{Document k}

Now, please remember the above content and answer the following question：
{query}

Figure 5: The prompt for response generation. The prompt is translated from Chinese.

You are a helpful AI assistant. You will see (1) the user query, (2) background information about the query, and (3) two different response A, B to the query. According to the background knowledge, you need to determine which response can better answer the query.

<Background Knowledge>
{DOC*}

<query>
{query}

A: $\{r_a\}$
B: $\{r_b\}$

Which option, A or B, is more effective for addressing the query stated above? Please consider this question thoughtfully, provide reasons to support your decision, and then ultimately select A or B.
Answer: Let's think step by step.

Figure 6: The prompt used for coherence evaluation. The prompt is translated from Chinese.

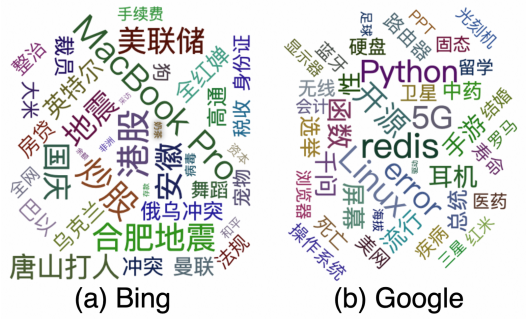


Figure 7: The Chinese visualization result of the high-frequency words.

Mikhail Yurochkin. 2023. Large language model routing with benchmark datasets. *arXiv preprint arXiv:2309.15789*.

Atsushi Sugiura and Oren Etzioni. 2000. Query routing for web search engines: Architecture and experiments. *Computer Networks*, 33(1-6):417–429.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *ArXiv*, abs/1409.3215.

Shuai Wang, Shengyao Zhuang, Bevan Koopman, and Guido Zuccon. 2024. Resllm: Large language models are strong resource selectors for federated search. *arXiv preprint arXiv:2401.17645*.

Yue Wang and Richong Zhang. 2018. A knowledge base question answering method based on dynamic programming. *CCKS(China Conference on Knowledge Graph and Semantic Computing)*.

Zhikun Xu, Yinghui Li, Ruixue Ding, Xinyu Wang, Boli Chen, Yong Jiang, Xiaodong Deng, Jianxin Ma, Hai-Tao Zheng, Wenlian Lu, Pengjun Xie, Chang Zhou, and Fei Huang. 2024. Let llms take on the latest challenges! a chinese dynamic question answering benchmark. *ArXiv*, abs/2402.19248.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Ke-Yang Chen, Kexin Yang, Mei Li, Min Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yunyang Wan, Yunfei Chu, Zeyu Cui, Zhenru Zhang, and Zhi-Wei Fan. 2024. Qwen2 technical report. *ArXiv*, abs/2407.10671.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *ArXiv*, abs/1904.09675.

A The Prompts in Our Method

Figure 5 and 6 show the prompts in response generation and coherence evaluation.

B Experimental Setting Details

Construction Details about PrivateQA PrivateQA primarily consists of questions related to timeliness. The question-answer examples are collected from public QA websites or translated from public English datasets using GPT4. We prioritize privacy protection and data integrity, ensuring that the dataset contains only open-domain QA content and excludes any private, harmful, or unethical material. PrivateQA is used to evaluate LLMs' capability to answer questions about timeliness. Due to confidentiality requirements, we are unable to disclose the dataset publicly.

	WebQA	PrivateQA	NLPCC-MH	CDQA	SogouQA
Number of samples	400	400	400	400	400

Table 5: The datasets statistics.

Data Statistics The statistics of used datasets are shown in Table 5. The examples are randomly sampled from the original data split.

C The Chinese Visualization Result

Figure 7 presents the original Chinese version of the high-frequency words.