

Article

An Expert Routing Method Based on Positive Sample Semantic Distribution via BGMM-LSE

Weifeng Ren ¹, Ting Zheng ¹, Jie Zhang ², Yunzhong Chen ², Erkang Wei ¹, Chenxiao Liu ³, Borui Fan ⁴, Zhaiyuan Ji ³, Yao Lu ^{3,*}, Jian He ³, Yaxin Gao ³ and Shanqing Yu ^{3,*}

¹ Institute of Public Security Science and Technology, Hangzhou 310009, China

² Department of Science and Technology and Information, Zhejiang Provincial Public Security Department, Hangzhou 310009, China

³ College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China

⁴ Hangzhou Dipont School of Arts and Science, Hangzhou 311100, China

* Correspondence: yaolu.zjut@gmail.com (Y.L.); yushanqing@zjut.edu.cn (S.Y.)

Abstract

Multi-expert large language model systems need to dynamically distribute user requests among multiple candidate experts to improve inference accuracy and system efficiency in complex task scenarios. To address the issues of existing routing methods relying on manual rules, unified discriminant boundaries, or large-scale annotated data, as well as high maintenance costs during expert expansion, this paper proposes an expert routing method, BGMM-LSE, based on positive sample semantic distribution modeling. The method constructs a positive sample set using only the requests historically successfully processed by each candidate expert, and maps the requests into dense semantic vectors through a pre-trained text feature extraction model; subsequently, a Bayesian Gaussian mixture model (BGMM) is independently trained for each expert to characterize the capability distribution of its successful requests in the semantic space. During online inference, the system encodes the target request into a feature vector, calculates the log-likelihood score combining the retained Gaussian component parameters and the smoothed covariance matrix of each expert, and uses Log-Sum-Exp to aggregate the probability contributions of different components. Finally, the request is routed to the expert model(s) with the highest score. Experiments were evaluated based on seven tasks: Math, GSM-Symbolic, HumanEval, Mbpp, MMLU, AIME1983–2025, and HellaSwag. The results show that this method achieves an average routing accuracy of 86.25%. The experimental results demonstrate that BGMM-LSE can provide stable request distribution capabilities for multi-expert large language model systems while maintaining interpretability and scalability.

Keywords: LLM routing; Bayesian Gaussian Mixture Model; large language models



Academic Editor: Sven Groppe

Received: 18 June 2026

Revised: 20 July 2026

Accepted: 22 July 2026

Published: 29 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article

distributed under the terms and

conditions of the [Creative Commons](#)

[Attribution \(CC BY\)](#) license.

1. Introduction

Large language models have achieved strong performance across a broad range of tasks, including mathematical reasoning, code generation, knowledge question-answering, and open-domain dialog. Nevertheless, in practical deployment, no single model consistently performs best across all input types. Different models often exhibit distinct strengths: some are more effective on programming tasks, some are more reliable in mathematical reasoning, and others are more competitive on broad knowledge or commonsense tasks. This heterogeneity has made multi-expert LLM systems an increasingly attractive deployment

paradigm, in which multiple candidate models are maintained and the system dynamically assigns each incoming request to an appropriate expert.

In such systems, the routing stage is critical because the overall system quality depends not only on the capability of the candidate experts, but also on whether an incoming request is assigned to a suitable one. Existing studies have approached this problem from several different perspectives. FrugalGPT formulates routing as part of a cost-aware cascade and aims to balance quality and budget across multiple LLM services [1]. RouteLLM learns routers from preference data and focuses primarily on selecting between stronger and weaker models under quality-cost trade-offs [2]. Hybrid LLM similarly emphasizes efficiency-aware query routing between models of different strengths [3]. RouterBench, on the other hand, provides a systematic benchmark for comparing routing strategies and shows that routing performance can vary substantially across methods and evaluation settings [4]. These works collectively demonstrate the importance of multi-LLM routing, but they also indicate that the formulation of the routing problem depends strongly on the target deployment scenario.

The scenario considered in this paper differs from the above cost-oriented or preference-oriented settings. Our focus is not on choosing between a single strong model and a single weak model, nor on optimizing a response-cost trade-off under a fixed budget. Instead, we study routing in a multi-expert setting where several candidate models possess partially overlapping but non-identical capabilities. In such a setting, a request may be answerable by more than one expert, and the practical objective is to assign it to the expert whose historical competence pattern best matches the semantic characteristics of the request. This makes the routing problem less naturally suited to a single global decision boundary.

A large portion of existing routing methods, including linear classifiers, multilayer perceptrons, graph-based routers, and nearest-neighbor or clustering-based approaches, can be viewed as learning or approximating a direct mapping from request representations to expert choices. These methods are effective in many settings and are also included in our empirical comparisons. However, they typically treat routing as a discriminative selection problem. In the multi-expert scenario studied here, such a formulation has two practical limitations. First, it does not explicitly model the capability region of each expert itself; instead, expert selection is determined through relative decision boundaries among experts. Second, when expert capabilities overlap or when new experts are added, the routing mechanism may require global adjustment or retraining. For systems that evolve over time, this increases maintenance complexity. In summary, the key gap addressed in this work is the lack of a routing framework that can model expert-specific capability regions directly from historical successful requests while naturally accommodating overlapping expert competencies.

Motivated by this observation, this paper investigates a different routing formulation: instead of directly learning a unified discriminative boundary across all experts, we model the semantic distribution of historically successful requests for each expert independently. The purpose of this study is not merely to improve routing accuracy, but to establish a probabilistic routing framework that is interpretable, modular, and easy to maintain in evolving multi-expert LLM systems. To this end, we propose BGMM-LSE (Supplementary Materials), an expert routing framework based on positive-sample semantic distribution modeling. The method first maps request texts into dense semantic vectors using a pre-trained embedding model, and then fits an expert-specific Bayesian Gaussian mixture model (BGMM) with a Dirichlet process prior over the successful requests of each expert. At inference time, a target request is routed by evaluating its log-likelihood under the capability distribution of each expert. To improve the numerical stability of

likelihood computation in high-dimensional semantic space, the proposed method introduces covariance smoothing to make component-wise covariance estimation more stable, and employs Log-Sum-Exp aggregation to avoid numerical underflow or overflow during the accumulation of mixture-component likelihoods.

The main contributions of this paper are summarized as follows:

- We formulate expert routing as an expert-wise semantic distribution matching problem and propose a routing framework that models each expert independently from its historical successful requests, rather than relying on a single global discriminative boundary.
- We introduce a Bayesian Gaussian mixture model with a Dirichlet process prior to characterize the multi-modal capability distribution of each expert, thereby allowing the effective mixture structure to adapt to the expert's positive-sample distribution without requiring a manually fixed number of active components.
- We develop a numerically stable likelihood-based routing mechanism by combining covariance regularization smoothing with Log-Sum-Exp aggregation, which improves the reliability of expert scoring in high-dimensional semantic space.

2. Related Work

2.1. Mixture of Experts and Routing Mechanisms

The basic idea of mixture of experts models can be traced back to the adaptive combination of local experts and hierarchical expert models. Jacobs et al. proposed the Adaptive Mixtures of Local Experts, which combines multiple local experts through a gating network [5]. Jordan and Jacobs further investigated the hierarchical mixture of experts architecture [6]. In the context of deep learning and large language models, sparse gating MoE expanded model capacity through conditional computation, and Sparsely Gated MoE demonstrated the potential of large-scale expert layers in language modeling and machine translation [7]. GShard combined conditional computation with automatic sharding for large-scale multilingual model training [8]; Switch Transformer reduced communication and training complexity through simpler Top-1 expert selection [9]. Subsequently, architectures like MMoE further emphasized the modeling of relationships between different tasks [10], and Expert Choice Routing improved expert utilization efficiency and load balancing from the perspective of routing strategies [11].

The above works primarily focus on the sparse activation of sub-network experts within the model. The objects faced by system-level candidate expert routing can be code assistants, legal assistants, general assistants, or external business models. Unlike internal MoE, system-level expert routing needs to face engineering constraints such as heterogeneous interfaces, the ability to add or remove experts, and the accumulation of historical execution results. Therefore, it is more suitable to adopt independent capability distribution modeling for experts, rather than training an end-to-end neural gating mechanism.

2.2. Multi-LLM Routing

Multi-LLM routing research focuses on dynamically selecting among multiple models to balance answer quality, cost, and efficiency. FrugalGPT reduces expensive model calls through a cascade calling strategy [1]. RouteLLM learns a dynamic router from preference data for quality–cost trade-offs [2]. Hybrid LLM selects between large and small models based on query difficulty and quality requirements [3]. RouterBench provides a systematic benchmark for comparing multi-LLM routing strategies [4]. LLM-Blender exploits complementary capabilities through candidate output ranking and fusion [12], whereas AutoMix decides whether to upgrade to a stronger model using the self-checking results of a smaller model [13].

More recently, Arch-Router maps requests to domain–action preference routes and supports the addition of candidate models without retraining the router [14]. Varangot-Reille et al. introduced Contextual Router, a retrieval-based meta-evaluation framework that estimates the performance of each candidate from similar historical requests [15]. Their K-NN Regression strategy selects a candidate according to the unweighted mean historical performance over retrieved neighbors, providing a lightweight non-parametric routing alternative.

Taken together, these studies demonstrate that request-level dynamic routing is vital for multi-model systems. Arch-Router relies on domain–action preference routes, whereas K-NN Regression relies on local performance retrieval. In contrast, the proposed method constructs a separate multi-modal probabilistic capability profile for each candidate expert from historically successful requests and routes a target request by likelihood matching. This formulation is designed for heterogeneous vertical experts and does not require predefined route labels.

Recent studies on LLM efficiency and collaborative model utilization further show that practical LLM systems increasingly rely on modular model management rather than a single monolithic model. For example, recent work has reassessed layer pruning strategies for efficient LLM deployment [16], explored collaborative annotation between large and small language models [17], and investigated progressive compression for efficient long-context reasoning [18]. These developments are complementary to routing research because they highlight the need for deployment-oriented mechanisms that can coordinate heterogeneous model capabilities under practical efficiency constraints.

2.3. Text Semantic Representation

Expert capability distribution modeling relies on stable text vector representations. Transformer established the foundation for modern pre-trained language models through self-attention mechanisms [19], and BERT improved representation capabilities for natural language understanding tasks through bidirectional pre-training [20]. Sentence-BERT transformed BERT into a dual-tower/triplet structure suitable for sentence vector similarity computation, which is widely used in semantic retrieval and clustering scenarios [21]. In recent years, contrastive pre-training has further enhanced general text representation capabilities. E5 and contrastive embedding methods targeting the joint space of text and code have both shown that high-quality vector representations can better support retrieval, matching, and clustering tasks [22,23]. The BGE model family provides general-purpose dense text representations for semantic retrieval and matching tasks [24]. The experimental implementation uses BAAI/bge-large-en-v1.5 to map request texts into 1024-dimensional dense feature vectors for subsequent expert capability distribution fitting.

2.4. Gaussian Mixture Models and Bayesian Nonparametric Modeling

Gaussian mixture models can represent complex sample distributions as a weighted combination of multiple Gaussian components, and parameter estimation is usually associated with the EM algorithm [25]. Bishop systematically compiled Gaussian mixture models, probability density estimation, and Bayesian modeling [26]. When the number of components is difficult to pre-determine manually, Dirichlet process mixture models and variational Bayesian inference provide a more flexible mixture modeling framework. Blei and Jordan provided a variational inference method for Dirichlet process mixture models [27], and Attias also discussed the variational Bayesian framework in graphical models relatively early [28]. Bayesian Gaussian mixture models can be used to characterize the multi-modal semantic distribution of positive samples for each expert, and combine

low-weight component filtering and covariance smoothing to form an expert capability profile that can be used for online routing.

3. Problem Definition

The multi-expert request routing task consists of candidate experts, historical requests, expert execution results, and online target requests. Suppose there are M candidate experts in the system. The candidate expert set is denoted as

$$\mathcal{E} = \{e_1, e_2, \dots, e_M\}, \quad (1)$$

where e_m represents the m -th candidate expert, and $m \in \{1, 2, \dots, M\}$ represents the expert index. The historical request set is denoted as

$$\mathcal{Q} = \{q_1, q_2, \dots, q_N\}, \quad (2)$$

where N represents the number of historical requests, q_i represents the i -th historical request text, and $i \in \{1, 2, \dots, N\}$ represents the request index.

Each historical request q_i corresponds to the execution results of multiple candidate experts. A binary variable $y_{i,m}$ is introduced to represent the execution result of expert e_m on request q_i : when $y_{i,m} = 1$, it means expert e_m successfully processed request q_i ; when $y_{i,m} = 0$, it means expert e_m failed to process request q_i . The positive sample set of the m -th expert is denoted as

$$\mathcal{D}_m^+ = \{q_i \mid y_{i,m} = 1, q_i \in \mathcal{Q}\}, \quad (3)$$

where $\mathcal{D}_m^+$ contains all the request texts successfully processed by expert e_m . If a request satisfies

$$\sum_{m=1}^M y_{i,m} = 0, \quad (4)$$

it indicates that the request failed to be processed by all candidate experts. This request is defined as a global invalid sample and is excluded during the training phase, participating in no expert capability distribution fitting.

4. Method

4.1. Overall Idea

The method consists of two phases: offline modeling and online routing. In the offline phase, it first extracts the requests successfully processed by each expert from the historical request texts and candidate expert execution results to construct the expert positive sample sets; subsequently, it performs text cleaning, tokenization, truncation, and format conversion on the positive sample request texts, and uses a pre-trained text feature extraction model to obtain dense feature vectors; finally, a Bayesian Gaussian mixture model is individually trained for each expert to extract and save the component weights, mean matrices, and covariance matrices, forming the expert capability profile.

In the online phase, upon receiving a new target request q^* , the system uses the same text feature extraction model to convert it into a target feature vector x^* . Subsequently, the system calculates the log-likelihood score of x^* under the capability distribution of each candidate expert, and obtains a stable total expert score through covariance smoothing and Log-Sum-Exp aggregation. Finally, the router compares the total scores of all candidate experts and forwards the request to the expert with the highest score; if the system strategy allows multi-expert collaboration, the request can also be forwarded to the multiple highest-scoring experts.

4.2. Text Preprocessing and Feature Encoding

For any request text q_i , a pre-trained text feature extraction model $f(\cdot)$ maps it to a d -dimensional dense feature vector x_i

$$x_i = f(q_i), \quad x_i \in \mathbb{R}^d. \quad (5)$$

where d represents the feature vector dimension. The experimental implementation used the pre-trained BAAI/bge-large-en-v1.5 model to map the positive sample texts of each expert into 1024-dimensional dense feature vectors to facilitate subsequent expert capability distribution fitting and routing calculations.

Text preprocessing includes four steps. First, text cleaning, to remove special symbols, meaningless characters, extra spaces, and line breaks. Second, tokenization and truncation, processing the text according to the tokenization rules of the selected pre-trained model, and truncating or segmenting texts that exceed the maximum input length of the model. Third, format conversion, converting the text into an input format acceptable to the model. Fourth, vector storage, saving the x_i corresponding to each request text associated with the original request text and the respective expert, forming the data structure needed for subsequent modeling.

For expert e_m , its positive sample vector set is defined as

$$X_m = \{x_i \mid q_i \in \mathcal{D}_m^+\}. \quad (6)$$

where X_m only contains the request vectors successfully processed by expert e_m , and does not contain the negative samples of other experts.

4.3. Expert Capability Distribution Modeling Based on BGMM

After completing the positive sample vectorization, the modeling input for candidate expert e_m is its exclusive positive sample vector set X_m . This set only contains the request vectors historically successfully processed by this expert, without introducing the positive samples of other experts as negative samples, nor constructing a unified global classification boundary. In this way, the capability of each expert no longer relies on discriminant boundaries with other experts but is characterized by the high-density regions of its historically successful requests in the semantic space. This modeling method transforms the global boundary problem into a description of the positive sample capability comfort zone, thereby avoiding the decision conflicts brought about by forced boundary division when the capabilities of different experts overlap.

To characterize the multi-modal structure in the positive sample semantic space, a Bayesian Gaussian mixture model (BGMM) is individually trained for each candidate expert. Let $K_{\max}$ be the preset maximum number of mixture components. Here, a component refers to a local Gaussian distribution fitted by BGMM in the expert's positive sample distribution, which is a semantic sub-region in the expert's capability space. A candidate expert usually excels not just at a single type of request, but may simultaneously cover multiple similar yet unidentical request clusters. The component weight reflects the proportion of this semantic sub-region in the expert's historically successful samples; the mean parameter characterizes the semantic center of this sub-region, and the covariance matrix characterizes the dispersion degree and feature correlation of the samples within this sub-region.

BGMM is based on a Gaussian mixture model and introduces a Dirichlet process prior in the mixture weight modeling. Through prior constraints, the model can adaptively determine the final effective number of retained components within the preset maximum component number $K_{\max}$, based on the distribution of expert e_m 's positive sample vectors in the feature space. The system first sets an upper limit of component number $K_{\max}$ for each expert, and then updates the weight, mean parameters, and covariance matrix of each component based on Bayesian inference during training. For components with a lower

match to the sample distribution, their weights will gradually decrease, and they will be treated as redundant components and filtered out after the model converges. Therefore, the model does not require manual assignment of a fixed number of components for different experts, which reduces modeling bias caused by improperly setting the number of components and adapts to the inconsistencies in the scale and semantic coverage of positive samples among different experts [26–28]. The capability distribution of expert e_m is expressed as

$$p(x | e_m) = \sum_{k=1}^{K_m} \pi_{m,k} \mathcal{N}(x | \mu_{m,k}, \Sigma_{m,k}), \quad (7)$$

where $x \in \mathbb{R}^d$ represents an arbitrary request feature vector, K_m represents the effective number of components finally retained by expert e_m , with $K_m \leq K_{\max}$, $k \in \{1, 2, \dots, K_m\}$ represents the component index, $\mathcal{N}(x | \mu_{m,k}, \Sigma_{m,k})$ represents the multivariate Gaussian probability density term with $\mu_{m,k}$ as the mean parameter and $\Sigma_{m,k}$ as the covariance matrix, and $\pi_{m,k}$, $\mu_{m,k}$, and $\Sigma_{m,k}$, respectively, represent the weight, mean parameter, and original covariance matrix of the k -th component for expert e_m . The component weights satisfy

$$\sum_{k=1}^{K_m} \pi_{m,k} = 1, \quad \pi_{m,k} \geq 0. \quad (8)$$

After the model converges, the system extracts the weight, mean parameter, and original covariance matrix of each component, and sets a preset weight threshold τ . When the weight of a certain component is lower than τ , this component usually only covers a small number of samples or noise regions, and the system regards it as a redundant component and filters it out. The weights of the remaining effective components are renormalized and saved as the expert capability profile. This process can effectively reduce the interference of low-weight noise components on online likelihood scoring, making the expert capability profile more centrally reflect the historically successful request regions. When a new candidate expert is added, it is only necessary to collect the successful requests of that expert and fit a new BGMM capability distribution, without needing to reconstruct the distribution parameters of other experts, nor retrain a unified classifier.

4.4. Covariance Smoothing

In the expert capability distribution modeling phase, the semantic capability space of each candidate expert is jointly characterized by several effective Gaussian components. Each Gaussian component consists of a component weight, a mean matrix, and a covariance matrix, where the covariance matrix is used to characterize the dispersion degree and correlation structure of the internal feature vectors in different dimensions for that component. Since text semantic vectors usually have high dimensions, and there are differences in the positive sample scale and semantic coverage among various experts, directly calculating the multivariate Gaussian probability density based on the original covariance matrix may lead to unstable covariance estimation, excessively large matrix condition numbers, or near-singularities, thereby affecting the numerical reliability of the log-likelihood score.

To mitigate the above issues, a covariance regularization smoothing mechanism is introduced during the expert score calculation phase. For every Gaussian component retained by a candidate expert, a regularized diagonal matrix is first constructed based on the average variance information estimated from its original covariance matrix; subsequently, this regularized diagonal matrix is added to the original covariance matrix to obtain the smoothed covariance matrix. This process is equivalent to introducing an isotropic stabilizing term based on the original covariance estimation, making the covariance matrix more numerically invertible and computationally stable while preserving its main distribution structure.

In the online routing phase, the smoothed covariance matrix is used to calculate the multivariate Gaussian probability density for the target feature vector. Given a new target request, the system first maps it to a target feature vector, and then, respectively, calculates the log-likelihood values of this vector under different Gaussian components of each candidate expert. Compared to directly using the original covariance matrix, the regularized smoothed covariance matrix can reduce the impact of local sample sparsity, concentrated feature distributions, or covariance degeneration in the high-dimensional semantic space on probability density estimation, thereby enhancing the stability of component-level likelihood calculations and providing a reliable numerical foundation for subsequent Log-Sum-Exp aggregated scoring.

4.5. Log-Sum-Exp Expert Score

After completing the candidate expert capability distribution modeling and covariance regularization smoothing processing, the system enters the online routing inference phase. Given a new target request q^* , it is first mapped to a fixed-dimensional target feature vector through the text feature extraction function $f(\cdot)$

$$x^* = f(q^*), \quad x^* \in \mathbb{R}^d. \quad (9)$$

where d represents the feature vector dimension, used to characterize the position of the target request in the semantic space, serving as the input for subsequent expert matching calculations.

For candidate expert e_m , its capability distribution is jointly represented by several effective Gaussian components retained during the training phase. Suppose this expert contains K_m effective components, where the weight, mean matrix, and smoothed covariance matrix of the k -th component are $\pi_{m,k}$, $\mu_{m,k}$, and $\Sigma_{reg,m,k}$, respectively. Then, the log-likelihood score of the target feature vector x^* under the capability distribution of expert e_m is expressed as

$$\log P(x^* | e_m) = \log \left(\sum_{k=1}^{K_m} \pi_{m,k} \mathcal{N}(x^* | \mu_{m,k}, \Sigma_{reg,m,k}) \right). \quad (10)$$

Here, $\mathcal{N}(\cdot)$ represents the multivariate Gaussian probability density function. The expression characterizes the matching relationship between the target feature vector x^* and the capability distribution of expert e_m from the perspective of a probabilistic generative model. By weighting and aggregating the probability densities of multiple effective Gaussian components, the marginal likelihood estimation of the target request in that expert's capability space is obtained. Compared to single-center representations or a single Gaussian assumption, the mixture Gaussian form can describe the multi-cluster distribution structure that the expert positive samples might have in the high-dimensional semantic space, thereby more fully reflecting the local heterogeneity and multi-modal characteristics of the expert's capability coverage.

To facilitate a unified comparison, the final score of expert e_m for the target request q^* is defined as

$$S_m = \log P(x^* | e_m). \quad (11)$$

In actual computational processes, since the multivariate Gaussian probability density in a high-dimensional semantic space may yield extremely small numerical values, directly performing weighted summation in the probability space is prone to numerical underflow or loss of precision. Therefore, the Log-Sum-Exp aggregation concept is adopted to stably complete the aggregation of each component's probability contribution within the logarithmic space. This processing not only maintains the probabilistic semantics of the Gaussian

mixture model but also improves the numerical stability of the expert score calculation in the high-dimensional feature space.

The system calculates the log-likelihood scores for all candidate experts respectively and determines the target routing expert based on the maximum score principle

$$e^* = \arg \max_{1 \leq m \leq M} S_m. \quad (12)$$

4.6. Algorithm Flow

Given a new target request text q^* , where $\star$ indicates a new input in the online phase, the router needs to select one or more target experts from $\mathcal{E}$. If only a single expert is selected, the target expert is denoted as e^* . The goal of routing is to make e^* match the semantic requirements of q^* and the historical successful capability distribution of the expert as much as possible. Detailed pipeline is presented in Algorithm 1. The key point of Algorithm 1 is that each expert is represented by an independently fitted positive-sample capability distribution, and routing is performed according to the likelihood of the target request under these expert-specific distributions. BGMM captures multiple semantic regions of successful expert behavior, while covariance smoothing and Log-Sum-Exp aggregation provide a stable scoring mechanism in the high-dimensional embedding space. The complete offline profile construction and online routing procedure is illustrated in Figure 1.

Algorithm 1: Expert Routing Algorithm Based on BGMM-LSE.

Input : $\mathcal{Q}$ (Historical request set), $\{y_{i,m}\}$ (Expert execution results), $\mathcal{E} = \{e_1, e_2, \dots, e_M\}$ (Candidate expert set), $f(\cdot)$ (Text feature extraction model), q^* (Target request)

Output: e^* (Target expert) or $\mathcal{E}^*$ (Target expert set), a^* (Answer returned by the expert)

- 1 **Offline Phase: Expert Capability Distribution Modeling**
- 2 $\mathcal{Q} \leftarrow \mathcal{Q} \setminus \{q_i \in \mathcal{Q} \mid \sum_{m=1}^M y_{i,m} = 0\}$ // Remove global invalid samples
- 3 **for** $m = 1$ **to** M **do**
- 4 $\mathcal{D}_m^+ \leftarrow \{q_i \in \mathcal{Q} \mid y_{i,m} = 1\}$ // Construct expert positive sample set
- 5 $\mathcal{D}_m^+ \leftarrow$ Text cleaning, tokenization truncation, and format conversion
- 6 $X_m \leftarrow \{f(q_i) \mid q_i \in \mathcal{D}_m^+\}$ // Vectorize positive samples
- 7 $p(x \mid e_m) \leftarrow \sum_{k=1}^{K_m} \pi_{m,k} \mathcal{N}(x \mid \mu_{m,k}, \Sigma_{m,k})$ // Estimate BGMM capability distribution of the expert
- 8 $\{\pi_{m,k}, \mu_{m,k}, \Sigma_{m,k}\}_{k=1}^{K_m} \leftarrow \{\pi_{m,k}, \mu_{m,k}, \Sigma_{m,k} \mid \pi_{m,k} \geq \tau\}, \sum_{k=1}^{K_m} \pi_{m,k} = 1$ // Extract effective components after convergence
- 9 Save the capability distribution parameters of expert e_m
- 10 **Online Phase: Target Request Routing**
- 11 $x^* \leftarrow f(q^*)$ // Vectorize target request
- 12 **for** $m = 1$ **to** M **do**
- 13 Read $\{\pi_{m,k}, \mu_{m,k}, \Sigma_{m,k}\}_{k=1}^{K_m}$
- 14 $\Sigma_{m,k} \rightarrow \Sigma_{reg,m,k}$ // Covariance regularization smoothing
- 15 $S_m \leftarrow \log P(x^* \mid e_m) = \log \left(\sum_{k=1}^{K_m} \pi_{m,k} \mathcal{N}(x^* \mid \mu_{m,k}, \Sigma_{reg,m,k}) \right)$ // Calculate expert log-likelihood score
- 16 $e^* \leftarrow \arg \max_{e_m \in \mathcal{E}} S_m$ // Determine target expert
- 17 **if** There are multiple highest-scoring experts or configured to return multiple experts **then**
- 18 $\mathcal{E}^* \leftarrow \emptyset$ // Initialize target expert set
- 19 **for** $m = 1$ **to** M **do**
- 20 **if** S_m satisfies preset return conditions **then**
- 21 $\mathcal{E}^* \leftarrow \mathcal{E}^* \cup \{e_m\}$ // Add to target expert set
- 22 **return** e^* or $\mathcal{E}^*, a^*$

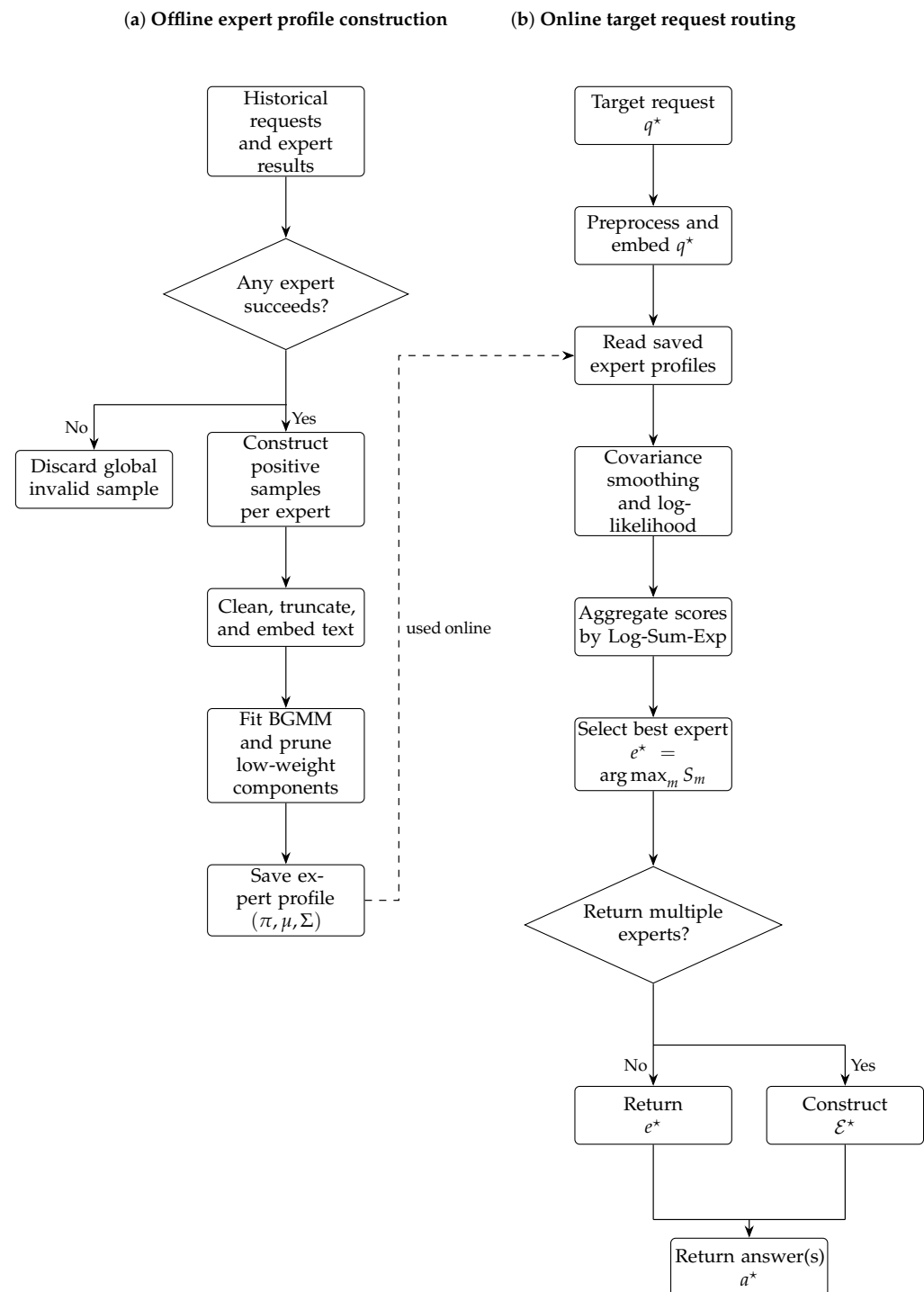


Figure 1. Activity diagram of BGMM-LSE, showing offline expert profile construction and online target request routing.

5. Experimental Setup

The experimental setup includes application scenarios, datasets, and evaluation metrics. It seamlessly connects with the method section, aiming to verify the effectiveness of the proposed BGMM-LSE expert routing method in multi-task scenarios. Specifically, by comparing the matching relationship between the request semantics and the distribution of the experts' historical successful samples, the experiments evaluate whether the method can accurately distribute different types of requests to suitable candidate experts, and analyze its routing performance on different datasets based on unified evaluation metrics.

5.1. Experimental Scenarios and Candidate Expert Models

The experimental scenario is configured as a request routing task in a multi-expert large language model question-answering system. The system receives natural language questions and, based on the semantic features of the requests and the historical performance of candidate experts, distributes the questions to the most suitable expert models for inference. To verify the method's effectiveness, this paper constructs multi-expert routing experiments on public benchmark datasets. The candidate experts cover coding, comprehensive knowledge, and commonsense reasoning capabilities. Training and testing data are constructed utilizing the answering results of each expert on the samples to further evaluate the dynamic distribution capability of the router.

The selected candidate models collectively represent the expected expert capability range in a multi-expert system. DeepSeek-R1-Distill-Qwen-14B represents reasoning-oriented capability for mathematical problem solving, logical analysis, and complex problem solving; Llama-3.3-70B-Instruct represents large-scale general instruction following, multilingual understanding, and text generation capability; Qwen3-30B-A3B represents balanced general reasoning and instruction-following capability; Qwen3-30B-A3B-Instruct-2507 represents complex instruction handling, tool use, and long-context understanding capability; Qwen3-Coder-30B-A3B-Instruct represents code generation, program understanding, and software engineering-related capability; and Qwen2.5-7B-Instruct represents lightweight general instruction-processing capability. The candidate set therefore includes both specialized and general-purpose experts, while retaining partial overlap in general reasoning and instruction following. Other available models were not selected because they did not provide a sufficiently distinct complementary role beyond the capability dimensions represented by the selected candidate models.

DeepSeek-R1-Distill-Qwen-14B was selected as a reasoning-oriented expert because of its focus on mathematical, coding, and complex logical reasoning; its limitation is that it is not primarily positioned as a broad general-assistance or vertical-domain model. Llama-3.3-70B-Instruct was selected as a large-scale general instruction expert because it provides broad coverage of language understanding, instruction following, text generation, and multilingual interaction; its limitation is that it is not specifically optimized for a single task type such as coding or mathematical reasoning. Qwen3-30B-A3B was selected as a general reasoning expert because it combines reasoning, instruction following, multilingual processing, and agent-style interaction; its limitation is that it does not serve as a dedicated code-specialized or long-context-specialized expert in the candidate set. Qwen3-30B-A3B-Instruct-2507 was selected as an instruction-optimized expert because it is suitable for complex multi-step instructions, logical reasoning, tool use, and long-context understanding; its limitation is that it remains a general instruction model rather than a specialist for a particular vertical domain. Qwen3-Coder-30B-A3B-Instruct was selected as a code-oriented expert because it targets code generation, program understanding, debugging, and tool use; its limitation is that its capability emphasis is more concentrated on software engineering and programming-related requests. Qwen2.5-7B-Instruct was selected as a lightweight general instruction expert because it can provide general instruction following, basic question answering, and reasoning with lower computational and deployment requirements; its limitation is that its smaller scale may constrain performance on complex reasoning, long-context, or highly specialized tasks relative to larger or more specialized models.

5.2. Dataset Setup

To more comprehensively evaluate the router's capabilities across different task types, the experiments divide the public task benchmarks into mathematics, coding, and reasoning categories, and construct test request sets using seven datasets. These datasets provide

heterogeneous request semantics and different degrees of expert overlap, which are suitable for evaluating whether the proposed router can assign requests to appropriate candidate experts across diverse task scenarios.

The seven datasets are summarized as follows:

- **Math:** A mathematical reasoning dataset used to evaluate the router's ability to handle algebra, geometry, and mathematical word problems [29];
- **GSM-Symbolic:** A symbolic mathematical reasoning dataset used to evaluate routing performance on formula derivation, symbolic calculation, and arithmetic reasoning requests [30];
- **HumanEval:** A code generation and program understanding dataset corresponding to coding-oriented requests [31];
- **Mbpp:** A basic programming problem dataset used to evaluate routing capability for function implementation, algorithm writing, and code completion tasks [32];
- **MMLU:** A multi-disciplinary benchmark covering broad knowledge comprehension, commonsense question-answering, and multi-domain reasoning [33];
- **AIME1983–2025:** A competition-level mathematical reasoning dataset used to evaluate routing effectiveness for high-difficulty mathematical problem-solving tasks;
- **HellaSwag:** A commonsense reasoning dataset used to evaluate open-ended context understanding and sentence completion tasks [34].

All datasets are converted into the same routing format. The text in the question field is used as the router input, while the `router_` columns provide binary expert correctness labels. Specifically, for expert e_m , $y_{i,m} = 1$ indicates that e_m correctly answers request q_i , and $y_{i,m} = 0$ otherwise. Other dataset-specific fields, such as choices, answers, reference solutions, or test information, are not used as router input. Before vectorization, the code converts each non-empty question value into a stripped string, and the embedding tokenizer truncates inputs longer than its maximum sequence length.

The data split is performed at the unique-question level. For each dataset, duplicate question strings are mapped to a single request entry, and the unique requests are divided into training and test sets with an 8:2 ratio using random seed 42. This prevents the same request text from appearing in both training and testing. During training, requests for which all candidate experts fail are excluded because they cannot provide positive samples for any expert profile. Each expert-specific BGMM profile is then fitted only with the positive samples of that expert.

5.3. Evaluation Metrics

For each test request q_i , the router returns a ranked list of candidate experts from $\mathcal{E}$. For top-1 routing, a result is considered correct when the selected expert can correctly answer the request according to the expert correctness label $y_{i,m}$. Let N_{test} denote the number of test samples, and let $\mathbb{I}_i$ denote whether the top-1 selected expert is correct. The top-1 routing accuracy is defined as

$$Acc = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \mathbb{I}_i. \quad (13)$$

For top- k evaluation, let $\hat{\mathcal{E}}_i^{@k} \subseteq \mathcal{E}$ denote the set of top- k experts returned for request q_i . Precision@ k , Recall@ k , F1@ k , and Hit@ k are computed as

$$\text{Precision@}k = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \frac{\sum_{e_m \in \hat{\mathcal{E}}_i^{@k}} y_{i,m}}{k}, \quad (14)$$

$$\text{Recall@}k = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \begin{cases} \frac{\sum_{e_m \in \hat{\mathcal{E}}_i^{\text{@}k} y_{i,m}}}{\sum_{m=1}^M y_{i,m}}, & \sum_{m=1}^M y_{i,m} > 0, \\ 0, & \sum_{m=1}^M y_{i,m} = 0, \end{cases} \quad (15)$$

$$\text{F1@}k = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \begin{cases} \frac{2 \sum_{e_m \in \hat{\mathcal{E}}_i^{\text{@}k} y_{i,m}}}{k + \sum_{m=1}^M y_{i,m}}, & \sum_{m=1}^M y_{i,m} > 0, \\ 0, & \sum_{m=1}^M y_{i,m} = 0, \end{cases} \quad (16)$$

$$\text{Hit@}k = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \mathbb{I} \left(\sum_{e_m \in \hat{\mathcal{E}}_i^{\text{@}k} y_{i,m} > 0 \right). \quad (17)$$

5.4. Computing Environment

All experiments were conducted on a Linux server equipped with eight NVIDIA Tesla V100 GPUs. The server was used to run the candidate expert models and the proposed BGMM-LSE routing framework. For large expert models such as Llama-3.3-70B-Instruct, the available multi-GPU resources were allocated according to the memory requirements of model inference.

6. Experimental Results

6.1. Overall Routing Performance

Table 1 presents the routing accuracy comparison on seven datasets. BGMM-LSE achieves the highest macro-average accuracy of 86.25% among the compared methods. On mathematical tasks, it obtains 85.16% on Math and 96.40% on GSM-Symbolic, while K-NN Regression is slightly higher on these two datasets. On coding tasks, BGMM-LSE matches the best HumanEval accuracy of 93.94% and achieves the best Mbpp accuracy of 79.00%. On reasoning tasks, it obtains the best results on MMLU (87.89%) and HellaSwag (79.75%), and matches the best AIME1983–2025 result of 81.63%. Overall, these results show that BGMM-LSE maintains competitive performance across diverse task types while achieving the strongest average routing performance.

Table 1. Performance comparison of different routing methods on mathematics, coding, and reasoning tasks.

Method\Dataset	Math		Coding		Reasoning			Average
	Math	GSM-Symb.	HumanEval	Mbpp	MMLU	AIME	HellaSwag	
Random	79.08%	90.00%	93.94%	68.00%	83.35%	65.31%	72.05%	78.79%
Random_Oracle_0.5	79.79%	89.17%	82.33%	75.16%	81.43%	59.07%	73.76%	77.24%
PRknn	76.00%	91.90%	78.79%	68.00%	76.36%	67.35%	65.90%	74.90%
LinearR	79.76%	91.30%	90.91%	72.00%	82.38%	60.71%	69.85%	78.13%
MLPR	77.52%	92.70%	84.85%	72.00%	77.32%	62.24%	65.20%	75.98%
C-Bert	70.29%	88.20%	66.67%	54.08%	73.37%	23.60%	67.53%	63.39%
MLC	78.98%	92.68%	77.78%	65.31%	79.03%	73.03%	66.19%	76.14%
GraphRouter	66.66%	88.74%	85.98%	57.31%	74.16%	23.80%	59.11%	65.11%
ArchRouter	76.00%	92.00%	90.91%	68.00%	85.08%	77.55%	79.25%	81.26%
K-NN Regression	85.28%	96.70%	90.91%	73.00%	86.66%	81.63%	78.90%	84.73%
Select the best expert	76.92%	89.80%	93.94%	68.00%	82.66%	68.37%	72.05%	78.82%
Ours	85.16%	96.40%	93.94%	79.00%	87.89%	81.63%	79.75%	86.25%

Table 2 further reports supplementary top- k routing metrics beyond the accuracy comparison in Table 1. Compared with both selecting the best expert and K-NN Regression, BGMM-LSE achieves higher Precision@1, Recall@1, F1@1, Hit@2, and Recall@2. These results indicate that BGMM-LSE improves both top-1 expert selection and the coverage of feasible experts in the top-2 ranked list.

Table 2. Supplementary top- k routing performance comparison beyond accuracy.

Method	Precision@1	Recall@1	F1@1	Hit@2	Recall@2
Select the best expert	84.12%	41.00%	53.35%	90.22%	76.80%
K-NN Regression	84.73%	41.17%	53.65%	90.73%	77.29%
Ours	86.25%	42.59%	55.10%	90.96%	78.16%

6.2. Computational Complexity and Memory Analysis

Table 3 shows that BGMM-LSE incurs an additional offline profile-construction cost of 8.90 s and a one-time profile-preparation cost of 0.10 s, whereas K-NN Regression requires only 0.08 s to construct its retrieval state. However, BGMM-LSE reduces the macro-average router-state memory from 19.25 MB to 15.44 MB and achieves substantially lower online routing latency, with mean and P95 values of 0.30 ms and 0.33 ms, respectively, compared with 8.23 ms and 13.20 ms for K-NN Regression. This trade-off arises because K-NN Regression retrieves from historical request embeddings at inference time, whereas BGMM-LSE performs likelihood scoring using compact precomputed expert profiles.

Table 3. Empirical resource comparison between K-NN Regression and BGMM-LSE. All values are macro-averaged over seven dataset configurations with the same data splits, candidate expert pools, and precomputed text embeddings. Online latency excludes text encoding.

Method	Offline Construction (s)	Profile Preparation (s)	Router State (MB)	Mean Online (ms)	P95 Online (ms)
K-NN Regression ($k = 40$)	0.08	–	19.25	8.23	13.20
BGMM-LSE (Ours)	8.90	0.10	15.44	0.30	0.33

6.3. Ablation Study

Table 4 reports the ablation results under the same per-dataset tuned setting as the main accuracy comparison. Removing the multi-component modeling structure reduces the average accuracy from 86.25% to 85.36%, indicating that representing each expert with multiple semantic regions contributes to routing performance. Removing covariance smoothing further reduces the accuracy to 84.76%, showing that regularized covariance estimation improves the reliability of likelihood-based scoring in the high-dimensional embedding space. Replacing Log-Sum-Exp aggregation with maximum component scoring does not change the average top-1 accuracy in this setting, suggesting that the dominant component often determines the final selected expert. Nevertheless, Log-Sum-Exp is retained because it provides a principled and numerically stable aggregation of mixture-component likelihoods.

Table 4. Ablation study.

Method	Accuracy	Change
Full BGMM-LSE	86.25%	–
w/o multi-component modeling	85.36%	−0.89
w/o covariance smoothing	84.76%	−1.50
w/o Log-Sum-Exp	86.25%	0.00

6.4. Routing Behavior and Case Analysis

Figure 2 shows that the distribution of selected experts varies across datasets. Rather than relying on a single globally preferred expert, BGMM-LSE assigns different proportions of requests to different experts according to their learned capability profiles. This task-dependent selection behavior is consistent with the heterogeneous strengths of the candidate experts.

Figure 3 presents representative routing cases from the test sets. The examples illustrate that the router selects experts according to the semantic match between a request and the learned profile of each expert, rather than applying a fixed expert-selection rule to all requests.

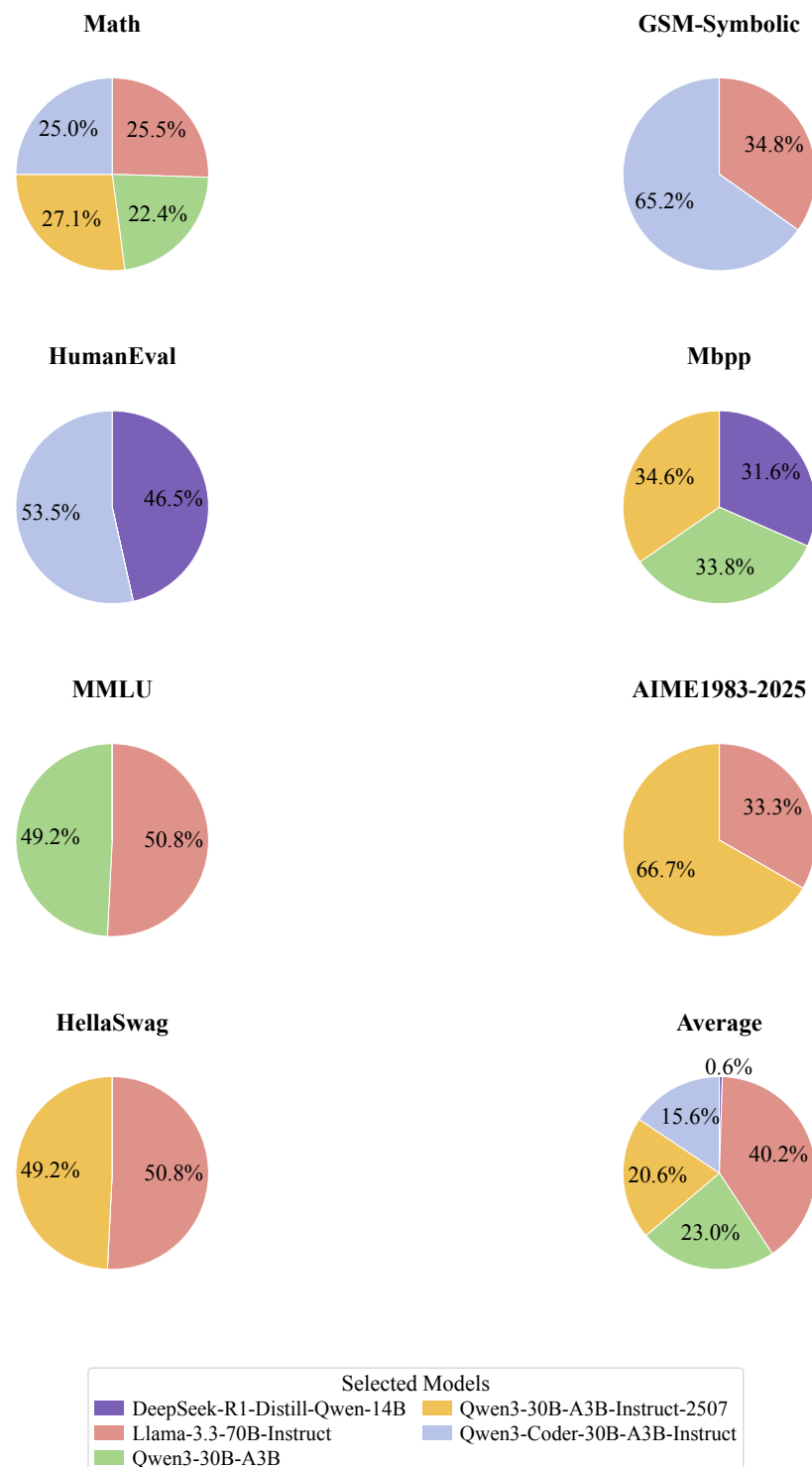


Figure 2. Proportion distribution of selected expert models on different datasets.

Overall, the experimental results support the effectiveness of BGMM-LSE for multi-task expert routing. By independently modeling the semantic distribution of historically successful requests for each candidate expert, the proposed method achieves the highest macro-average accuracy of 86.25% across the seven datasets. The supplementary metrics, ablation results, resource analysis, and routing cases further characterize its ranking quality, component contributions, practical scale, and task-dependent routing behavior.

[Question] He scrubs in the shampoo and then washes it off. He then combs it and blow dries his hair after styling it with gel.
 [Options] A. then rinses it off in the sink.
 B. lets his hair down, giving it a final blow dry before putting it next to his face.
 C. uses an electric clipper to groom the sideburns and the temples.

Expert DeepSeek-R1-Distill-Qwen-14B Answer: D ✗	Expert Llama-3.3-70B-Instruct Answer: C ✓	Expert Qwen3-30B-A3B Answer: D ✗
Expert Qwen3-30B-A3B-Instruct-2507 Answer: D ✗	Expert Qwen3-Coder-30B-A3B-Instruct Answer: D ✗	Ours Ours Router -> Llama-3.3-70B-Instruct Answer: C ✓

[Question] Statement 1 | If a group has an element of order 15, it must have at least 8 elements of order 15. Statement 2 | If a group has more than 8 elements of order 15, it must have at least 16 elements of order 15.
 [Options] A. True, True B. False, False C. True, False D. False, True

Expert DeepSeek-R1-Distill-Qwen-14B Answer: A ✓	Expert Llama-3.3-70B-Instruct Answer: B ✗	Expert Qwen3-30B-A3B Answer: A ✓
Expert Qwen3-30B-A3B-Instruct-2507 Answer: A ✓	Expert Qwen3-Coder-30B-A3B-Instruct Answer: C ✗	Ours Ours Router -> Qwen3-30B-A3B Answer: D ✓

Figure 3. Expert routing case analysis of the BGMM-LSE Router on real test requests.

7. Discussion

7.1. Consistency Between Method and Application Scenarios

The experiments use public benchmark datasets to construct multi-expert request routing scenarios to ensure the reproducibility of the results. It is important to note that while public benchmarks can simulate different types of user requests, they do not equate to the actual expert execution logs in a real system. In practical deployment, expert positive samples should still originate from real historical requests and the execution results of candidate experts.

7.2. Advantages of Positive Sample Modeling

The primary advantages of positive sample semantic distribution modeling lie in maintainability and interpretability. The capability profile of each expert is saved independently. When a new expert is added, it is only necessary to collect the successfully processed requests of that expert and fit a new BGMM, without the need to retrain a unified classifier for all experts.

7.3. Necessity of Covariance Smoothing and Log-Sum-Exp

In the 1024-dimensional text vector space, covariance matrices might become non-invertible due to insufficient samples, strong feature correlations, or extreme distributions. Covariance smoothing based on average variance adds an adaptive diagonal perturbation to the original matrix, which can improve matrix invertibility and the stability of high-dimensional probability calculations. Log-Sum-Exp ensures that the probabilistic aggregation of multiple Gaussian components stably completes in the logarithmic space, preventing numerical underflow caused by direct probability summation.

7.4. Discussion on the Choice of BGMM with a Dirichlet Process Prior

BGMM-DP is adopted because an expert's successful requests may occupy multiple semantic regions rather than a single unimodal region. It models each expert with local Gaussian components, while the Dirichlet process prior adaptively suppresses low-weight components during fitting. Although an upper bound $K_{\max}$ is specified, the effective mixture structure is determined by the positive-sample distribution. In our experiments, the tuned $K_{\max}$ is one or two for each dataset and is kept unchanged for all GMM-based methods.

Table 5 compares BGMM-DP-LSE with fixed-component and BIC-selected GMM routers under identical data splits and expert pools. BGMM-DP-LSE achieves 86.25% average accuracy, outperforming fixed-component GMM-LSE by 1.00 percentage point and BIC-selected GMM-LSE by 1.39 percentage points. BIC selects one component on average, suggesting that its complexity penalty favors an effectively unimodal model in the high-dimensional embedding space. Although BIC-selected GMM is marginally better on Math (85.24% versus 85.16%), BGMM-DP-LSE is equal or better on the remaining datasets and obtains the best overall average.

Table 5. Comparison with conventional GMM routing alternatives under the same data splits, expert pools, and per-dataset tuned component upper bounds. The reported accuracy is the macro-average over seven datasets.

Method	Component Selection	Accuracy (%)	Δ vs. Ours (pp)
BGMM-DP-LSE (Ours)	DP-BGMM	86.25	—
Fixed-component GMM-LSE	Fixed at the tuned upper bound	85.26	−1.00
BIC-selected GMM-LSE	BIC selection from 1 to the tuned upper bound	84.87	−1.39

BGMM-LSE has additional fitting and likelihood-computation costs compared with lightweight discriminative routers, and it assumes that expert capability regions can be approximated by Gaussian mixture components. Nevertheless, the results support the full BGMM-DP configuration as a more effective and interpretable probabilistic router than the two conventional GMM alternatives considered here.

7.5. Reusability in Other Use Cases

The proposed BGMM-LSE framework can be reused in other multi-expert decision scenarios where historical request records and expert execution outcomes are available. The method does not rely on task-specific model architectures or manually designed routing rules. Instead, it only requires three types of information: the request text, the candidate expert set, and the binary success label indicating whether each expert can correctly handle the request. Therefore, the same framework can be transferred to other domains by replacing the expert pool and collecting domain-specific historical execution records.

For example, in a multi-agent coding assistant, different experts may specialize in code generation, debugging, test generation, or repository-level understanding. BGMM-LSE can model the successful request distribution of each coding expert and route new programming requests accordingly. In a customer-service system, experts may correspond to different service departments or domain-specific dialog agents, and the router can learn capability profiles from historical resolved cases. In a medical or legal assistance system, the same framework can be used to route requests to experts specialized in different subdomains, provided that appropriate expert success labels are available.

When applied to a new use case, the main adaptation steps are straightforward. First, historical requests and the corresponding expert execution results are collected. Second, positive samples are extracted for each expert and embedded into the semantic space using the same text encoder or a domain-adapted encoder. Third, an expert-specific BGMM profile is fitted from the positive samples. After this offline construction stage, online requests can be routed by the same likelihood-based scoring and Log-Sum-Exp aggregation mechanism. If a new expert is added, only the capability profile of that expert needs to be fitted, while the existing expert profiles can be reused.

8. Conclusions

This paper proposes an expert routing method, BGMM-LSE, based on positive sample semantic distribution modeling, designed to solve the request distribution problem in heterogeneous multi-expert large language model systems. By modeling the semantic distribution of an expert's historical successful samples, the method achieves probabilistic matching between user requests and candidate expert capabilities, and reduces reliance on unified classification boundaries while maintaining interpretability and scalability. Experimental results indicate that the BGMM-LSE Router achieves an average accuracy of 86.25% across the seven datasets, verifying its effectiveness in multi-task routing scenarios encompassing mathematics, coding, and reasoning.

Supplementary Materials: The following supporting information can be downloaded at: <https://www.mdpi.com/article/10.3390/info17080736/s1>, The source code for the core BGMM-LSE algorithm has been provided as supplementary implementation material.

Author Contributions: Conceptualization, W.R., C.L. and S.Y.; methodology, W.R., C.L. and Z.J.; software, C.L., E.W. and J.H.; validation, T.Z., J.Z. and Y.C.; formal analysis, C.L. and Z.J.; investigation, W.R., Y.L. and S.Y.; resources, J.Z., Y.C. and Y.G.; data curation, T.Z., E.W. and J.H.; writing—original draft, W.R. and C.L.; writing—review and editing, Y.L., Y.G. and S.Y.; visualization, Z.J., J.H. and B.F.; supervision, Y.L., Y.G. and S.Y.; project administration, W.R. and S.Y.; funding acquisition, Y.G. and S.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key R&D Program of China (grant number 2025YFA1510900), the Baima Lake Laboratory Joint Fund of Zhejiang Provincial Natural Science Foundation of China (grant number LBMHZ25F020002), the Applied Basic Research Program of Zhejiang Province (grant number 2026C02A1264), the Yangtze River Delta Science and Technology Innovation Community Joint Research Project (grant numbers 2025CSJGG01000 and 2026ZY03003), the Key Research and Development Program of Zhejiang Province (grant number 2024C01025), the Key Technology Research and Development Program Project of Hangzhou (grant number 2024SZD1A23), and the Research and Development Program of Zhejiang Province (grant number 2026C02A1233).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets used in this study, namely Math [29], GSM-Symbolic [30], HumanEval [31], Mbbp [32], MMLU [33], AIME1983–2025, and HellaSwag [34], are publicly available. They can be accessed through open-source data repositories (e.g., <https://huggingface.co/datasets> (accessed on 10 July 2026)) or via the respective references cited in this manuscript.

Conflicts of Interest: The authors declare no conflicts of interest.

References

- Chen, L.; Zaharia, M.; Zou, J. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. *arXiv* **2023**, arXiv:2305.05176.
- Ong, I.; Almahairi, A.; Wu, V.; Chiang, W.-L.; Wu, T.; Gonzalez, J.E.; Kadous, M.W.; Stoica, I. RouteLLM: Learning to Route LLMs with Preference Data. In *Proceedings of the International Conference on Learning Representations*; ICLR: Dublin, Ireland, 2025.
- Ding, D.; Mallick, A.; Wang, C.; Sim, R.; Mukherjee, S.; Ruhle, V.; Lakshmanan, L.V.S.; Awadallah, A.H. Hybrid LLM: Cost-efficient and Quality-aware Query Routing. In *Proceedings of the International Conference on Learning Representations*; ICLR: Dublin, Ireland, 2024.
- Hu, Q.J.; Bieker, J.; Li, X.; Jiang, N.; Keigwin, B.; Ranganath, G.; Keutzer, K.; Upadhyay, S.K. RouterBench: A Benchmark for Multi-LLM Routing System. *arXiv* **2024**, arXiv:2403.12031.
- Jacobs, R.A.; Jordan, M.I.; Nowlan, S.J.; Hinton, G.E. Adaptive Mixtures of Local Experts. *Neural Comput.* **1991**, *3*, 79–87. [[CrossRef](#)] [[PubMed](#)]
- Jordan, M.I.; Jacobs, R.A. Hierarchical Mixtures of Experts and the EM Algorithm. *Neural Comput.* **1994**, *6*, 181–214. [[CrossRef](#)]
- Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; Dean, J. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *Proceedings of the International Conference on Learning Representations*; ICLR: Dublin, Ireland, 2017.
- Lepikhin, D.; Lee, H.; Xu, Y.; Chen, D.; Firat, O.; Huang, Y.; Krikun, M.; Shazeer, N.; Chen, Z. GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding. In *Proceedings of the International Conference on Learning Representations*; ICLR: Dublin, Ireland, 2021.
- Fedus, W.; Zoph, B.; Shazeer, N. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *J. Mach. Learn. Res.* **2022**, *23*, 5232–5270.
- Ma, J.; Zhao, Z.; Yi, X.; Chen, J.; Hong, L.; Chi, E.H. Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*; Association for Computing Machinery: New York, NY, USA, 2018; pp. 1930–1939.
- Zhou, Y.; Lei, T.; Liu, H.; Du, N.; Huang, Y.; Zhao, V.Y.; Dai, A.M.; Chen, Z.; Le, Q.V.; Laudon, J. Mixture-of-Experts with Expert Choice Routing. In *NIPS'22: Proceedings of the 36th International Conference on Neural Information Processing Systems*; Curran Associates Inc.: Red Hook, NY, USA, 2022; Volume 35, pp. 7103–7114.
- Jiang, D.; Ren, X.; Lin, B.Y. LLM-Blender: Ensembling Large Language Models with Pairwise Ranking and Generative Fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2023.
- Aggarwal, P.; Madaan, A.; Anand, A.; Potharaju, S.P.; Mishra, S.; Zhou, P.; Gupta, A.; Rajagopal, D.; Kappaganthu, K.; Yang, Y.; et al. AutoMix: Automatically Mixing Language Models. In *NIPS '24: Proceedings of the 38th International Conference on Neural Information Processing Systems*; Curran Associates Inc.: Red Hook, NY, USA, 2024.
- Tran, C.; Paracha, S.; Hafeez, A.; Chen, S. Arch-Router: Aligning LLM Routing with Human Preferences. *arXiv* **2025**, arXiv:2506.16655.

15. Varangot-Reille, C.; Bouvard, C.; Gourru, A. Generalising LLM Routing using Past Performance Retrieval: A Few-Shot Router is Sufficient. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2026; pp. 304–319.
16. Lu, Y.; Cheng, H.; Fang, Y.; Wang, Z.; Wei, J.; Xu, D.; Xuan, Q.; Zhu, Z. Reassessing Layer Pruning in LLMs: New Insights and Methods. In *Proceedings of the Fourteenth International Conference on Learning Representations*; ICLR: Dublin, Ireland, 2026.
17. Lu, Y.; Ji, Z.; Du, J.; Yu, S.; Xuan, Q.; Zhou, J.T. AutoAnnotator: A Collaborative Annotation Framework for Large and Small Language Models. *Trans. Mach. Learn. Res.* **2025**, *12*, 18–22.
18. Gao, Y.; Lu, Y.; Zhang, Z.; Nie, J.; Yu, S.; Xuan, Q. DSPC: Dual-Stage Progressive Compression Framework for Efficient Long-Context Reasoning. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*; IEEE: New York, NY, USA, 2026.
19. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. In *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*; Curran Associates Inc.: Red Hook, NY, USA, 2017.
20. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019.
21. Reimers, N.; Gurevych, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of EMNLP-IJCNLP*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019.
22. Wang, L.; Yang, N.; Huang, X.; Jiao, B.; Yang, L.; Jiang, D.; Majumder, R.; Wei, F. Text Embeddings by Weakly-Supervised Contrastive Pre-training. *arXiv* **2022**, arXiv:2212.03533.
23. Neelakantan, A.; Xu, T.; Puri, R.; Radford, A.; Han, J.M.; Tworek, J.; Yuan, Q.; Tezak, N.; Kim, J.W.; Hallacy, C.; et al. Text and Code Embeddings by Contrastive Pre-Training. *arXiv* **2022**, arXiv:2201.10005.
24. Xiao, S.; Liu, Z.; Zhang, P.; Muennighoff, N. C-Pack: Packaged Resources To Advance General Chinese Embedding. In *Findings of the Association for Computational Linguistics: ACL 2023*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2023; pp. 7463–7473.
25. Dempster, A.P.; Laird, N.M.; Rubin, D.B. Maximum Likelihood from Incomplete Data via the EM Algorithm. *J. R. Stat. Soc. Ser. B* **1977**, *39*, 1–38. [[CrossRef](#)]
26. Bishop, C.M. *Pattern Recognition and Machine Learning*; Springer: New York, NY, USA, 2006.
27. Blei, D.M.; Jordan, M.I. Variational Inference for Dirichlet Process Mixtures. *Bayesian Anal.* **2006**, *1*, 121–143. [[CrossRef](#)]
28. Attias, H. A Variational Bayesian Framework for Graphical Models. In *NIPS'99: Proceedings of the 13th International Conference on Neural Information Processing Systems*; MIT Press: St. Cambridge, MA, USA, 2000.
29. Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; Steinhardt, J. Measuring Mathematical Problem Solving With the MATH Dataset. *arXiv* **2021**, arXiv:2103.03874.
30. Mirzadeh, I.; Alizadeh, K.; Shahrokhi, H.; Tuzel, O.; Bengio, S.; Farajtabar, M. GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models. *arXiv* **2024**, arXiv:2410.05229.
31. Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; Pinto, H.P.O.; Kaplan, J.; Edwards, H.; Burda, Y.; Joseph, N.; Brockman, G.; et al. Evaluating Large Language Models Trained on Code. *arXiv* **2021**, arXiv:2107.03374.
32. Austin, J.; Odena, A.; Nye, M.; Bosma, M.; Michalewski, H.; Dohan, D.; Jiang, E.; Cai, C.; Terry, M.; Le, Q.; et al. Program Synthesis with Large Language Models. *arXiv* **2021**, arXiv:2108.07732.
33. Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; Steinhardt, J. Measuring Massive Multitask Language Understanding. In *Proceedings of the International Conference on Learning Representations*; ICLR: Dublin, Ireland, 2021.
34. Zellers, R.; Holtzman, A.; Bisk, Y.; Farhadi, A.; Choi, Y. HellaSwag: Can a Machine Really Finish Your Sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.