Guarded Query Routing for Large Language Models

Richard Šléher[†a], William Brach[†a,*], Tibor Sloboda[a,c,d], Kristián Košťál[a] and Lukas Galke[b]

[a]STU, Bratislava, Slovakia
[b]Centre for Machine Learning, University of Southern Denmark
[c]aleph0 s.r.o.
[d]NetFire LLC
ORCID (Richard Šléher[†]): https://orcid.org/0009-0000-0069-969X, ORCID (William Brach[†]): https://orcid.org/0009-0002-0321-0321, ORCID (Tibor Sloboda): https://orcid.org/0000-0001-6817-6297, ORCID (Kristián Košťál): https://orcid.org/0000-0003-0679-4588, ORCID (Lukas Galke): https://orcid.org/0000-0001-6124-1092

Abstract. Query routing, the task to route user queries to different large language model (LLM) endpoints, can be considered as a text classification problem. However, out-of-distribution queries must be handled properly, as those could be questions about unrelated domains, queries in other languages, or even contain unsafe text. Here, we thus study a *guarded* query routing problem, for which we first introduce the Guarded Query Routing Benchmark (GQR-Bench), which covers three exemplary target domains (law, finance, and healthcare), and seven datasets to test robustness against out-of-distribution queries. We then use GQR-Bench to contrast the effectiveness and efficiency of LLM-based routing mechanisms (GPT-4o-mini, Llama-3.2-3B, and Llama-3.1-8B), standard LLM-based guardrail approaches (LlamaGuard and NVIDIA NeMo Guardrails), continuous bag-of-words classifiers (WideMLP, fastText), and traditional machine learning models (SVM, XGBoost). Our results show that WideMLP, enhanced with out-of-domain detection capabilities, yields the best trade-off between accuracy (88%) and speed (<4ms). The embedding-based fastText excels at speed (<1ms) with acceptable accuracy (80%), whereas LLMs yield the highest accuracy (91%) but are comparatively slow (62ms for local Llama-3.1:8B and 669ms for remote GPT-4o-mini calls). Our findings challenge the automatic reliance on LLMs for (guarded) query routing and provide concrete recommendations for practical applications. GQR-Bench will be released as a Python package — `gqr`. Source code: https://github.com/williambrach/gqr.

1 Introduction

Large Language Models (LLMs) are responding to millions of queries each day across a wide range of applications. To save computational resources, it is of high interest to route each query to LLMs of appropriate sizes, inference compute budgets, and domain specialization. We view this query routing problem as an instance of text classification, where the task is to classify a query into a set of suitable endpoints (e.g., LLM agents, rule-based chatbots, human chat partners). See Figure 1. While such a routing mechanism can be prepared for in-distribution queries (e.g., known languages, a pre-defined set of target domains, generally safe inputs), it must also be able to handle out-of-distribution queries (out of domain, unsupported languages, unsafe inputs). We denote this task *guarded* query routing.

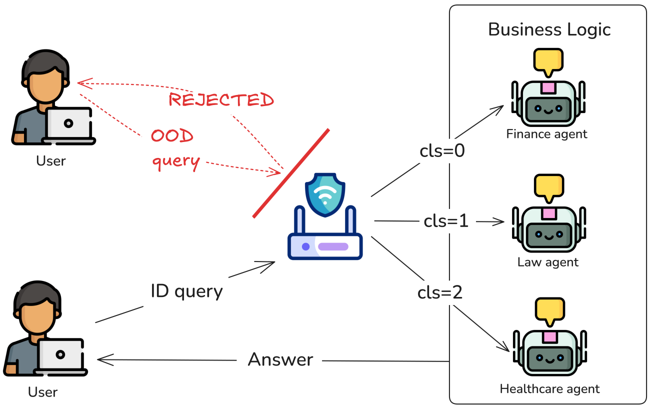

Figure 1. A guarded query routing system that efficiently filters out-of-domain (OOD) queries while directing valid in-domain (ID) queries to appropriate domain-specific LLMs.

Guarded query routing is an extremely challenging task because the models cannot be prepared for all possible out-of-distribution queries. This implies that we must assume that there is no training data available for out-of-distribution queries (otherwise, they would not be out of distribution). Presumably for this reason, practitioners currently rely on LLMs to carry out the routing, such as NVIDIA NeMo Guardrails [37] or LlamaGuard [20]. However, LLM calls come with extensive computational costs, and we hypothesize that efficient text classifiers combined with an out-of-distribution detection mechanism would perform similarly well with less compute.

To test our hypothesis, we first set up a new evaluation benchmark, called Guarded Query Routing Benchmark (GQR-Bench), that simulates a guarded query routing problem. GQR-Bench is composed of three datasets that we consider our target domains (law, finance, health), as well as seven datasets from which we draw different types of out-of-distribution queries: out-of-domain queries, queries in different languages, and queries with unsafe inputs. We further provide

[*] Corresponding author - william.brach@stuba.sk
[1] † - These authors contributed equally to this work.

a suggested evaluation methodology that balances in-domain accuracy and robustness against out-of-distribution queries.

Based on recent surveys in text classification [15], we then evaluate strong text classification methods such as BERT [9], fastText [23], and WideMLP [13] on GQR-Bench, testing their suitability as guarded query routers. We also test traditional text classification methods (e.g., SVM, XGBoost) on top of contextualized embeddings from a pre-trained language models. We test these methods against common strategies for standard query routing (e.g., routing via a strong LLM such as GPT-4o-mini) as well as standard guardrails to prevent unsafe inputs (e.g., NVIDIA NeMO Guardrails), which we consider as our baselines.

Our results show that efficient text classifiers are almost on par with the much more costly yet common approach of using an LLM as-a-router with WideMLP attaining 95% relative performance – while requiring orders of magnitude less time. Generally, WideMLP displays a promising trade-off between speed and accuracy, while fastText is even faster but with notable decrease in accuracy. In addition, we show that standard guardrail techniques and previous approaches to query routing do *not* perform well on GQR-Bench.

Generally, we find that established machine learning techniques can achieve comparable accuracy to LLMs in guarded query routing with dramatically less latency, in some cases to under 1 ms, and eliminating the costs associated with LLM calls. As such, this research challenges the automatic reliance on LLMs for query routing and provides practical alternatives for optimizing the performance-cost-latency triangle in domain-specific applications.

In summary, the main contributions of this paper are threefold:

1. We present a new benchmark, GQR-Bench, along with an evaluation methodology, for guarded query routing, where the routing mechanisms need to determine the correct domain, while being robust against out-of-distribution queries.
2. We run and evaluate several models on this benchmark, including large language models, efficient text classifiers, and traditional machine learning models based on embeddings – to test their suitability for guarded query routing.
3. Our results show that efficient text classifiers display good performance with notable improvements in latency, challenging common practice to employ LLMs for the task of query routing.

2 Related Work

The rationale behind query routing lies in its ability to improve cost efficiency while maintaining the integrity of responses [40]. This allows users to optimize the allocation of computational resources and operational expenses by directing less complex queries to smaller, more economical models, reserving more sophisticated models for tasks requiring advanced capabilities or rejecting queries identified as malicious or outside the domain of interest [18, 19, 43]. A primary application domain involves the systematic routing between models with different sizes and functional capabilities [33]. This dynamic allocation methodology enables systems to delegate less cognitively demanding tasks to models with fewer parameters, thereby preserving the computational resources of larger architectures for queries requiring enhanced precision or advanced reasoning capabilities. Studies using real-world examples show that this query routing can reduce the use of large models by up to 40% without affecting their quality [48, 10]. The theoretical foundations of query routing extend beyond mere model selection. It takes inspiration from conceptually adjacent frameworks, such as Mixture-of-Experts architectures [25, 42], which incorporate internal routing mechanisms, and Retrieval-Augmented Generation (RAG) systems. RAG systems employ contextual classification[1] methods to select appropriate document repositories and determine whether retrieval augmentation is necessary for specific queries [5]. Despite the advancements in this field, there are still many research challenges to overcome, such as enhancing router generalization across diverse model pairings and data distributions [26, 32]; developing task-aware routing mechanisms with contextual sensitivity [47, 10]; and establishing evaluation benchmarks designed specifically for testing routing performance. In summary, while query routing is a well-studied problem in different contexts, the crucial aspect of robustness to out-of-distribution queries has not been studied so far.

3 Problem Statement: Guarded Query Routing

We study the problem of guarded query routing, where a machine learning model has to decide to which sink an incoming query should be routed. Such sinks could include domain-specific LLM agents, a more powerful LLM, a rule-based system, or a human chat partner. However, practical implementations must account for completely unrelated queries that the routing system is not prepared for (like those in other languages, from different domains, or unsafe inputs). Handling these appropriately requires the system to be *guarded*.

This guarded query routing problem can be understood as a text classification problem that must be robust with respect to out-of-distribution examples. More formally, given a set of domains $D_1, D_2, \ldots, D_k$, the task is to classify a previously unseen query into one of the domains $\{1, 2, \ldots, k\}$, or conclude that the current query does not belong to any of the domains. Importantly, we assume that there are training data available for each domain D_i, but that there are no training data available for out-of-distribution queries, which reflects real-world exposure to uncontrolled queries.

This definition can be readily extended to other aspects, e.g., whether one of these 'domains' must be handled by human chat partners or a more powerful (bigger, more test-time compute) language model. However, for the scope of this paper, we deem it sufficient to study a limited set of target domains while introducing variety in the out-of-distribution queries to test the system's robustness.

4 Guarded Query Routing Benchmark (GQR-Bench)

The proposed evaluation framework, GQR-Bench, is composed of existing datasets - Table 1. Its primary purpose is to evaluate methods on the guarded query routing problem, as defined in the previous section. In the following, we will first lay out the employed datasets before we present our suggested evaluation methodology.

4.1 Dataset Composition

We consider three target domains (law, finance, health), each supported by a dedicated dataset. These datasets represent a domain part of the guarded query router implementation. In addition, we employ seven more datasets to form out-of-distribution queries in other domains, languages, and unsafe inputs.

[1] https://weaviate.io/developers/contributor-guide/contextionary/classification-benchmarks

Table 1. Number of examples in the datasets included in GQR-Bench

Dataset	#train	#valid	#test
Datasets for target domains (in-distribution)			
Law StackExchange Prompts	9611	2402	2987
Question-Answer Subject Finance Instruct	9635	2409	2956
Lavita ChatDoctor HealthCareMagic 100k	9554	2389	3057
Datasets for out-of-distribution queries			
Jigsaw	0	0	3214
OLID	0	0	860
HateXplain	0	0	5935
dk_hate	0	0	329
HateSpeech Slovak	0	0	959
Machine Learning	0	0	128
Web Questions	0	0	2032

In-distribution datasets for the target domains To cover the three target domains, law, finance, and healthcare, we choose Law StackExchange Prompts [29], Question-Answer Subject Finance Instruct [7], and Lavita ChatDoctor HealthCareMagic 100k [46]. These datasets serve as our in-domain datasets and can thereby be employed as a training set for guarded query routers. The goal is that the query router can only accept queries that are from the **law**, **finance**, or **healthcare** domains. The test split of these datasets will be used to assess the effectiveness of in-distribution routing.

Out-of-distribution Datasets As sources for out-of-distribution queries, we selected datasets that focus on toxicity (Jigsaw [22]), offensive language (OLID [51]), and hate speech (HateXplain [30]). For evaluation of capabilities to not answer queries in different languages, we employ datasets dk_hate [45] that contain offensive questions in Danish and the hate speech Slovak dataset[2]. To evaluate no offensive/hate domains, we also employ two Q&A datasets - Machine Learning[3] and Web Questions [2]. It is crucial that these out-of-distribution datasets are not available for training. Instead, the query routers only see queries from these datasets at test time.

4.2 *Evaluation methodology*

Our evaluation methodology is structured around three key performance metrics, each addressing a distinct aspect of query routing capabilities. Together, these metrics provide a complete evaluation of a model's suitability for deployment in real-world applications, where reliability, accuracy, and efficiency are critical concerns. **In-distribution (ID) accuracy** assesses the effectiveness of a model acting as a query router, quantifying its ability to accurately classify queries into appropriate categories, essentially measuring classification accuracy. Notably, queries that were rejected for classification are not counted as correct. **Out-of-distribution (OOD) accuracy** assesses whether the routing mechanism has correctly rejected an OOD query. This metric provides insight into the model's ability to protect systems against out-of-distribution queries.

GQR-Score: Harmonic Mean between ID and OOD Accuracy We adopt the harmonic mean as our primary evaluation metric to balance the model's performance on both in-domain (ID) and out-of-distribution (OOD) classification tasks. The harmonic mean provides a stringent measure of combined performance because it heavily penalizes imbalances between the two accuracy scores. Unlike the arithmetic mean, which can mask poor performance in one category when the other performs exceptionally well, the harmonic mean approaches zero if either component score approaches zero. Mathematically, given ID accuracy $\textsc{Acc}_{\text{ID}}$ and OOD accuracy $\textsc{Acc}_{\text{OOD}}$, the harmonic mean H is calculated as:

$$\text{GQR-Score} = \frac{2 \cdot \textsc{Acc}_{\text{ID}} \cdot \textsc{Acc}_{\text{OOD}}}{\textsc{Acc}_{\text{ID}} + \textsc{Acc}_{\text{OOD}}} \tag{1}$$

This harmonic mean resembles a joint score for ID and OOD performance, giving us a valuable signal of the overall performance in guarded query routing, where a model must correctly route ID queries and identify OOD queries. This ensures that the models must be able to tackle both classification tasks simultaneously to attain a high GQR-Score. In practical settings, this balanced measure is critical because failures in either dimension can affect the overall system's utility and user trust.

Efficiency Lastly, we consider latency and disk size efficiency metrics crucial for practical applications. Integrating high-latency models could impact inference response times and consequently degrade user experience.

We further report disk size as a robust metric comparable across different models and technical setups.

5 Guarded Query Routing Models

The selection of initial models to evaluate on GQR-Bench was guided by three key considerations: (1) computational efficiency for real-time routing applications, (2) architectural diversity ranging from decoder-only LLMs to contextualized embedding models combined with traditional text classifiers, and (3) established performance in text classification tasks.

5.1 *Baselines: LLMs and Embedding Similarity*

As our baselines, we consider approaches that are commonly employed in practical applications. This entails, most prominently, offloading the decision of how a query should be routed to a large language model itself. In addition, we look at popular guardrail techniques, as we would expect them to perform particularly well on out-of-distribution detection. Third, we include an open-source library for tackling the query routing task. This library can be combined with various embedding models and claims high accuracy and efficiency.

Large Language Models as Query Routers: We implemented guarded query routing mechanisms using various LLMs as preprocessing filters that evaluate requests before forwarding them to the primary model. Our evaluation included *GPT-4o-mini, Llama3.1-8B, and Llama3.2-3B*. The employed prompt template is provided in Supplementary Material A.

Nvidia NeMo Guardrails: This framework [37] functions as a protective guardrail for deployed LLMs by utilizing a pre-processing evaluation mechanism to assess input conformity with established safety parameters. We evaluated the NeMo guardrail system with multiple LLM gatekeepers—specifically *GPT-4o-mini, Llama3.1-8B, and Llama3.2-3B*. The guardrail configuration is provided in Supplementary Material A.

Llama-Guard: We tested two specialized safety models: Llama-Guard-3-8B and Llama-Guard-3-1B from the Llama 3 family [11]. These variants of the Llama architecture have undergone fine-tuning specifically for safety applications. The models execute content evaluation protocols and produce binary classifications indicating whether inputs comply with safety guidelines. While Llama-Guard models cannot function as guard query routers due to their lack of routing capabilities, we included them in our evaluation to benchmark the effectiveness of other models in safety prediction against these specialized safety models.

[2] https://huggingface.co/datasets/TUKE-KEMT/hate_speech_slovak
[3] https://huggingface.co/datasets/mjphayes/machine_learning_questions

Semantic Router[4]: This open-source library implements a vector space-based decision layer for routing LLM requests. In contrast to approaches that depend on computationally expensive LLM generations for classification decisions, Semantic Router leverages semantic embeddings for efficient routing determinations. The system defines "routes" (decision paths) using exemplar utterances, encoding these utterances into vector representations, and then comparing input queries against these vector spaces to identify the most appropriate routing path.

5.2 *Efficient Text Classifiers*

We employ a variety of text classification models and adapt them for the problem of guarded query routing.

WideMLP [13] has attained best performance in a comprehensive survey [15], even surpassing LLMs. WideMLP operates on bag-of-words input representations and employs a small number of wide hidden layers. In particular, we employed one hidden layer with 1024 hidden units. The initial hidden layer is implemented efficiently through an embedding table, and the pooling over the sequential dimension is weighted by the tokens' IDF scores (as in TF-IDF).

We adapted WideMLP for the guarded query router by making two key changes: First, we trained the model with binary cross-entropy instead of categorical cross-entropy to avoid overconfident mispredictions caused by the softmax. Second, we employed a threshold on the confidence scores to detect out-of-domain queries: When the predicted probability for all classes falls below this threshold, we consider the current example out-of-distribution. We systematically evaluated multiple threshold configurations (0.5, 0.7, 0.9, and 0.99) to determine the optimal balance between domain classification accuracy and out-of-domain detection. These changes are inspired by previous work on text classification under the presence of OOD examples [44], which further investigates class-specific thresholds obtained through a risk reduction technique. However, follow-up work has found that these class-specific thresholds obtained through risk reduction are not more effective than a correctly set global threshold [14] – which is why we do not employ risk reduction here.

fastText [4] has extended word2vec [31] by a fall-back for out-of-vocabulary words through bags of character n-grams, enabling effective handling of out-of-vocabulary terms. For classification, fastText employs pre-trained word embeddings and trains a logistic regression on top [23]. The pre-trained embeddings come in a large variety of languages, each trained mainly on the Wikipedia corpus of the respective language.

To adapt fastText for guarded query routing, we employed a One-vs-Rest approach, whereby we train one binary classifier per domain that produces a binary prediction (0 or 1) to predict domain membership. When all domain-specific classifiers return 0 for a given query, the system identifies it as out-of-distribution and rejects it accordingly. Hyperparameter selection was handled using fastText's autotune feature, with a time budget set to 300 seconds per classifier to find optimal settings based on validation performance.

Fine-tuned encoder-only language models We fine-tuned two encoder-only language models for query routing: *ModernBERT-base* and *BERT-base-multilingual-cased*. Both models were used with their default tokenizers without any modifications. The training process consisted of 5 epochs with learning rates between 2e-5 to 5e-5. These models were trained as multi-label classification problems; thus, binary cross-entropy (BCE) was used as the loss function. After obtaining predictions from the BERT models, a sigmoid activation function was applied to transform the output logits into probabilities within the 0-1 range. We then employed a similar thresholding approach as the WideMLP classifier to determine if a query belongs to a specific domain class or should be considered out-of-distribution.

5.3 *Sentence Embeddings with Traditional Classifiers*

We also draw traditional text classifiers into the comparison, motivated by their good performance in simple text categorization tasks [14]. Specifically, we employ these classifiers in conjunction with different contextualized embeddings. These included dense embedding architectures such as *all-MiniLM-L6-v2*, *all-MiniLM-L12-v2* [38], and *BGE-small-en-v1.5* [50]. We also evaluated sparse embedding methods, such as operating on TF-IDF weighted bag-of-words features [39, 8].

All traditional text classifiers are extended to be suitable for guarded query routing through a One-vs-Rest mechanism – fitting one model for each of the target domains, and concluding that a specific input query is out-of-distribution if none of the classifiers predict respective class membership. We choose the one with the highest prediction if multiple classes are positive.

XGBoost [6]: A gradient boosting framework with strong performance across diverse machine learning tasks [3]. We selected XGBoost for its robustness in handling sparse input features and effectiveness with high-dimensional text representations. We implemented a One-vs-Rest classification strategy, training separate binary classifiers for each domain to maximize classification precision while providing natural support for out-of-domain detection through confidence scoring mechanisms. This approach enables effective domain boundary discrimination even when dealing with semantically similar queries across multiple domains.

Support Vector Machines [16]: A well-established approach for out-of-distribution prediction that has demonstrated strong performance in question classification [35] and short text classification scenarios [1]. SVMs excel at finding optimal decision boundaries in high-dimensional feature spaces, making them particularly suitable for query routing applications where input texts are brief but semantically dense. We paired SVM with TF-IDF vectorization [12, 36] to enhance its ability to identify and categorize domain-specific linguistic patterns. Our implementation follows a One-vs-Rest architecture, which supports multi-domain classification and provides an elegant mechanism for detecting out-of-domain queries through margin-based confidence estimation.

MLP on Sentence Embeddings We have further experimented with a custom MLP model applied to pre-trained sentence embeddings. For this, we attached an MLP classification head directly to the embedding model.

The custom classification head incorporates recent advancements. First, we employ DynamicTanh (DyT) [28] as an alternative to layer normalization.

Second, we employ a customized SwiGLU activation function and associated layer structure [41], which progressively downsamples the embedding dimension to match the number of target classification labels (output logits). For regularization, we employed AlphaDropout [24] to retain normalization of the sentence embedding input and the DyT modules. We further use Focal Loss [27] due to its effectiveness in handling class imbalance.

For training, we use a two-phase fine-tuning strategy, a common practice in transfer learning [17]. In the first phase, the pre-trained embedding model's weights were frozen, and only the classification

[4] https://github.com/aurelio-labs/semantic-router

head was trained using a relatively high learning rate. In the second phase, both the classification head and the embedding model were trained concurrently (unfrozen), but their learning rates decreased by an order of magnitude. This process included an initial learning rate warm-up period, starting two orders of magnitude lower and gradually increasing. This warm-up phase is beneficial, particularly for stabilizing training in transformer-based architectures [49]. For efficiency, we converted the final model to the ONNX format [34] and applied INT8 quantization [21]. For OOD detection, we employ a threshold of 0.99.

6 Results

We first present the results for effectiveness, distinguished by in-distribution and out-of-distribution accuracy. We then present the results on the combined GQR score, before we analyze efficiency with respect to latency and disk size, and study the crucial efficiency-effectiveness trade-off.

In-Distribution Accuracy Table 2 presents the performance of different query routing approaches on GQR-Bench, with GQR-Score as our primary evaluation metric, as it penalizes models that excel in one dimension while performing poorly in the other.

Among large language models, Llama3.1-8B demonstrates the strongest balanced performance with a harmonic mean of 91.67%, closely followed by GPT-4o-mini at 91.61%. Interestingly, Llama3.2-3B exhibits an unusual behavior with exceptional OOD accuracy (99.82%) but poor ID accuracy (26.37%), resulting in a substantially lower harmonic mean of 41.72%. This suggests Llama3.2-3B may be overly conservative, frequently classifying even in-domain queries as out-of-distribution. The Semantic Router approaches using pre-trained embeddings show moderate performance, with all-MiniLM-L6-v2 achieving a harmonic mean of 57.69%. While these models maintain strong ID accuracy (approximately 90%), their OOD detection capabilities are limited, with accuracies ranging from 35.39% to 42.45%. This indicates that embedding similarity may be insufficient for OOD detection.

Among the continuous bag-of-words approaches, TF-IDF with WideMLP using a confidence threshold of 0.99 achieves the highest harmonic mean (87.74%), approaching the performance of LLM-based routers. This model demonstrates a well-balanced performance with 84.49% ID accuracy and 91.25% OOD accuracy. The fastText model also performs admirably with a harmonic mean of 80.12%, showing strong ID accuracy (95.80%) but moderate OOD accuracy (68.85%). Our experiments with various confidence thresholds for WideMLP reveal an interesting trade-off: as the threshold increases from 0.75 to 0.99, ID accuracy decreases from 93.67% to 84.49%, while OOD accuracy improves substantially from 73.33% to 91.25%. This demonstrates how threshold tuning can balance the requirements of specific deployment scenarios.

Traditional machine learning approaches also show competitive performance. The BGE-small-en-v1.5 embedding combined with an SVM using RBF kernel achieves the highest harmonic mean in this category (82.94%) with exceptional ID accuracy (99.42%) and respectable OOD accuracy (71.15%). Similarly, the XGBoost classifier with the same embeddings performs well with a harmonic mean of 81.04%. Interestingly, TF-IDF with SVM or XGBoost shows a reversed pattern compared to other models, with relatively low ID accuracy but strong OOD performance.

Table 2. Main Results. In-distribution (ID) accuracy, out-of-distribution (OOD) accuracy, and GQR-Score as combined score between ID and OOD accuracy. Best methods per category of approaches highlighted in bold. The highest GQR-Score is achieved by Llama3.1:8b, closely followed by GPT-4o-mini and then WideMLP.

Model	**ID Acc.**	**OOD Acc.**	**GQR-Score**
Baselines: Routing based on large language models			
Llama3.1:8b	95.66	**88.00**	**91.67**
GPT-4o-mini	**95.70**	87.85	91.61
Llama3.2:3b	26.37	99.82	41.72
Baselines: Embedding similarity approaches			
all-MiniLM-L6-v2 + S.R.	90.00	**42.45**	**57.69**
bge-small-en-v1.5 + S.R.	**90.70**	35.39	50.91
Continuous bag-of-words classifiers			
fastText	**95.80**	68.85	80.12
WideMLP(t=0.99)	84.49	**91.25**	**87.74**
WideMLP(t=0.90)	90.91	80.39	85.33
WideMLP(t=0.75)	93.67	73.33	82.26
Fine-tuned encoder-only language models			
ModernBERT-base (t=0.99)	**99.94**	19.74	32.97
BERT-base (t=0.99)	99.90	**19.80**	**33.05**
Sentence embeddings + traditional classifiers			
bge-small-en-v1.5 + SVM	**99.42**	71.15	**82.94**
bge-small-en-v1.5 + XGBoost	98.78	68.70	81.04
all-MiniLM-L6-v2 + SVM	86.06	63.14	72.84
all-MiniLM-L6-v2 + XGBoost	92.93	68.73	79.02
all-MiniLM-L12-v2 + MLP	95.17	59.60	73.23
TF-IDF + XGBoost	34.76	**84.49**	49.26
TF–IDF + SVM	34.36	55.31	42.39

Guardrail performance on in-domain queries While our primary focus is on models that can perform complete query routing, we also evaluated the ability of the guardrail models to identify in-domain queries as safe for processing correctly. Table 3 presents binary accuracy results for these guardrail models on the in-domain test set. The NeMo Guardrails framework shows varying performance depending on the underlying LLM. With Llama3.2-8B, it achieves an impressive 99.04% accuracy in correctly identifying in-domain queries as safe for processing. The same framework with Llama3.2-3B maintains strong performance at 91.80%, while the implementation with GPT-4o-mini shows a substantial drop to 72.57%. This variation suggests that the effectiveness of guardrail frameworks is heavily dependent on the capabilities of the underlying model. Surprisingly, the specialized Llama-Guard models, which are fine-tuned explicitly for content safety evaluation, perform poorly at identifying legitimate in-domain queries as safe. Llama-Guard-3-1B achieves only 34.21% accuracy, while Llama-Guard-3-8B performs even worse at 22.56%.

Table 3. Binary accuracy of guardrail models on ID test set. These guardrail models are not able to fully categorize the queries.

Model	**Accuracy**
NeMo Guardrails + Llama3.2:3B	91.80
NeMo Guardrails + Llama3.2:8B	99.04
NeMo Guardrails + GPT-4o-mini	72.57
Llama-Guard-3-1B	34.21
Llama-Guard-3-8B	22.56
cls-Llama-Guard-3-1B	0.04
cls-Llama-Guard-3-8B	0.00

Out-of-Distribution Detection Table 4 provides a breakdown of OOD performance across the different datasets, allowing us to assess model robustness to various types of out-of-distribution queries.

Despite being specifically designed for safety detection, Llama-Guard models demonstrate surprisingly weak performance on our OOD datasets, with average accuracies of only 22.56% and 34.21%

for the 8B and 1B variants, respectively. This suggests specialized safety models may not generalize well to the broader query routing task. Llama3.2-3B shows remarkable consistency across all OOD datasets, maintaining near-perfect detection rates. In contrast, Llama3.1-8B and GPT-4o-mini show strong performance on most datasets but struggle with ML questions (46.09% and 45.31% respectively). This suggests these models may have difficulty distinguishing between legitimate domain queries and out-of-domain but non-toxic queries. When used with NeMo Guardrails, GPT-4o-mini performs well (88.11% average), but both Llama variants show substantially reduced effectiveness (20.95% and 42.12%). This unexpected degradation indicates that guardrail frameworks may interfere with the models' inherent capabilities for domain distinction.

The WideMLP model with a 0.99 confidence threshold demonstrates remarkable consistency across all OOD datasets, ranging from 80.60% to 99.16% accuracy. This consistent performance across diverse datasets suggests strong generalization capabilities. FastText shows more moderate but balanced performance across datasets (54.46% to 83.11%). Traditional classifiers exhibit interesting patterns across datasets. TF-IDF with XGBoost achieves near-perfect accuracy on specific datasets (dkhate, TUKE SK, Web Q, ML Q) but struggles with others (Jigsaw, OLID, HateXplain), suggesting sensitivity to dataset characteristics. SVM and XGBoost display a balanced but moderate performance across datasets.

GQR-Score Among the LLM-based approaches, Llama3.1-8B achieved the highest GQR-Score (91.67%), closely followed by GPT-4o-mini (91.61%). Despite Llama3.2-3B's exceptional OOD accuracy (99.82%), its poor ID performance resulted in a substantially lower GQR-Score (41.72%). In the continuous bag-of-words category, TF-IDF with WideMLP (t=0.99) demonstrated the strongest balanced performance (87.74%), approaching LLM-level effectiveness. Embedding similarity approaches with Semantic Router showed moderate results, with all-MiniLM-L6-v2 reaching 57.69%. Traditional ML methods proved competitive, with BGE-small-en-v1.5 combined with SVM achieving 82.94%. Fine-tuned encoder-only models performed poorly, with BERT-base-multilingual-cased reaching only 33.05%. Notably, WideMLP achieves 95.71% of the best LLM performance while requiring much less computational resources.

Efficiency-Effectiveness Tradeoff Table 6 provides a visual representation of the tradeoff between model GQR-Score and efficiency as measured by latency. The upper-left quadrant represents the ideal region of high performance and low latency. FastText and WideMLP attain GQR-Scores above 80% with sub-millisecond latencies. LLM-based approaches (4o-mini and Llama3.1-8B) achieve the highest GQR-Score, but are 2-3 orders of magnitude slower than fastText.

Latency Table 5 provides an overview of latency across several batch sizes. LLM-based approaches consistently exhibit the highest latencies. GPT-4o-mini shows the highest single-query latency (because of REST API call) at 0.667 seconds, while Llama models range from 0.042 to 0.083 seconds per query. Batch processing improves efficiency the LLMs. Semantic Router yields a moderate latency (0.035 seconds per query) with no improvements through batching. FastText shows exceptional efficiency with latencies of approximately 0.00007-0.00009 seconds per query, roughly three orders of magnitude faster than LLM approaches. SVM and XGBoost models paired with various embedding approaches show excellent efficiency. Specifically, the embedding-based XGBoost models benefit substantially from batch processing, improving from 0.022 seconds per query to as low as 0.00035 seconds per query. All experiments were conducted on a single server with specifications: 2× ASUS GeForce RTX 4090 GPUs, AMD Threadripper PRO 7965WX (24-Core) processor, and 256GB RAM.

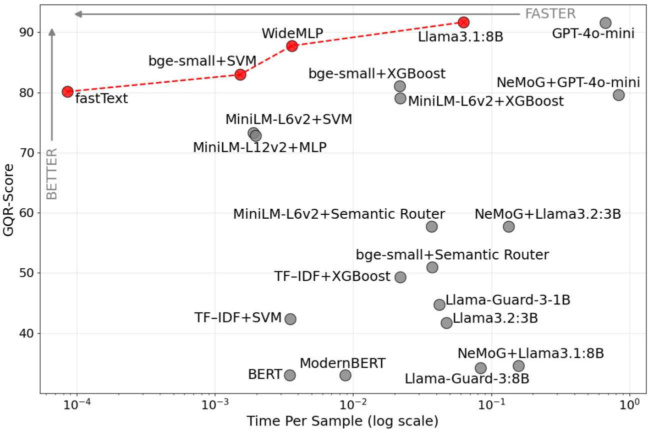

Figure 2. Efficiency-effectiveness tradeoff showing harmonic mean performance versus latency (log scale). Models in the upper-left quadrant offer optimal balance between high performance and low latency.

Disk size Model size on disk varies substantially across routing models (see Table 5), with important implications for deployment scenarios where storage and memory are constrained (e.g., edge devices). LLM-based approaches require the most storage, with Llama models ranging from approximately 2 GB to 4.9 GB. In contrast, traditional classification approaches based on TF-IDF require only between 4 and 11 MB. Models based on all-MiniLM-L12-v2 require about 100 MB, and the quantized MLP uses about 21.8 MB. Our one-vs-rest fastText approach uses 221 MB of disk size.

7 Discussion

Our key finding is that efficient text classifiers perform comparably to LLM-based approaches in guarded query routing at drastically lower latency. Specifically, we found that WideMLP with a high confidence threshold achieves 95.71% of the performance of the best LLM (Llama3.1-8B) at two orders of magnitude less latency – suggesting that robust and efficient text classifiers can be considered for query routing in practice.

Why combine expert routing and safety checks? Our proposed evaluation framework, GQR-Bench, integrates expert routing and safety checks. Both of these subtasks can be understood as text classification tasks. If the expert routing happens at the start of the pipeline, the router must also handle out-of-distribution queries, because they would otherwise cause overconfident mispredictions (when the classifier is trained via softmax). Furthermore, tackling both tasks together provides an opportunity to reduce computational costs – especially since standard guardrails (NeMO, LlamaGuard) are usually based on LLMs. While the use of LLMs taps into the extensive training that LLMs received, the safety question can also be handled by more efficient text classifiers combined with an OOD detection mechanism – as we have shown here.

Subpar performance of existing methods Despite being designed explicitly for safety, existing guardrail techniques displayed poor performance in GQR-Bench, as they are not well-suited for ID routing.

Routing based on embedding similarity is an interesting direction, as it requires only few examples per domain. However, SemanticRouter's GQR-Score was only moderate due to very limited OOD

Table 4. Breakdown of OOD accuracy across all considered datasets, and each model's unweighted average accuracy. The best average accuracy is highlighted per category of model. Manually prompted Llama3.2:3b performs best, followed by WideMLP.

Model	Jigsaw	OLID	HateXplain	dkhate	TUKE SK	Web Q	ML Q	Average
Baselines: Routing based on large language models								
Llama-Guard-3-8B	27.07	24.77	93.28	5.17	7.51	0.10	0.00	22.56
Llama-Guard-3-1B	51.40	61.40	91.47	12.77	20.13	2.31	0.00	34.21
Llama3.2:3b	99.69	99.88	99.98	100.00	100.00	99.16	100.00	**99.82**
Llama3.1:8b	94.43	93.60	97.99	95.74	97.60	90.55	46.09	88.00
NeMo Guardrails + Llama3.2:3B	61.42	59.65	43.15	61.09	67.88	1.67	0.00	42.12
NeMo Guardrails + Llama3.1:8B	51.99	36.40	20.83	10.33	27.11	0.00	0.00	20.95
NeMo Guardrails + GPT-4o-mini	98.26	94.19	99.78	91.49	96.14	57.19	79.69	88.11
GPT-4o-mini	94.71	93.49	98.10	94.53	98.02	90.80	45.31	87.85
Baselines: Embedding similarity approaches								
all-MiniLM-L6-v2 + Semantic Router (s=5, t=0.5)	22.96	31.74	36.71	39.51	20.33	96.70	49.22	**42.45**
bge-small-en-v1.5 + Semantic Router (s=5, t=0.5)	15.15	28.95	32.67	31.91	12.41	95.42	31.25	35.39
Continuous bag-of-words classifiers								
fastText	74.46	61.51	54.46	74.77	83.11	70.37	63.28	68.85
WideMLP (t=0.99)	93.83	93.49	91.00	86.93	80.60	99.16	93.75	**91.25**
WideMLP (t=0.90)	87.87	83.26	77.56	71.73	56.93	95.57	89.84	80.39
WideMLP (t=0.75)	84.04	76.74	70.48	57.45	47.34	92.91	84.38	73.33
Fine-tuned encoder-only language models								
ModernBERT-base (t=0.99)	27.10	17.91	18.06	10.33	2.50	62.30	0.00	19.74
BERT-base-multilingual-cased (t=0.99)	20.91	28.26	25.44	25.84	30.87	7.28	0.00	**19.80**
Sentence embeddings + traditional classifiers								
bge-small-en-v1.5 + SVM	77.47	75.00	63.81	61.40	63.82	59.69	96.88	71.15
bge-small-en-v1.5 + XGBoost	81.95	68.26	72.15	47.72	59.02	58.81	92.97	68.70
all-MiniLM-L6-v2 + SVM	59.61	71.74	61.63	37.99	34.62	81.89	94.53	63.14
all-MiniLM-L6-v2 + XGBoost	47.57	77.44	53.14	57.45	60.17	95.47	89.84	68.73
all-MiniLM-L12-v2 + MLP	74.77	80.47	85.59	56.23	18.87	68.45	32.81	59.60
TF–IDF + SVM	24.58	26.16	21.72	75.38	96.98	54.87	87.50	55.31
TF–IDF + XGBoost	58.31	67.44	66.40	100.00	99.90	99.36	100.00	**84.49**

Table 5. Model latency comparison (in seconds) for different batch sizes as quantified by the mean time taken per single query in a batch.

Model	Size on disk MB	Batch Size				
		1	32	64	128	256
Routing based on Large Language Models						
Llama-Guard-3-1B	3GB	0.04192	0.08678	0.18224	0.42820	1.08448
Llama3.2:3B	2GB	0.04713	0.01403	0.01835	0.02312	0.02022
Llama3.1:8B	4.9GB	0.06275	0.01970	0.02213	0.02566	0.02263
Llama-Guard-3-8B	4.9GB	0.08349	0.08112	0.08113	0.08118	0.08113
NeMo Guardrails + Llama3.2:3B	2GB	0.13303	0.12944	0.12968	0.12954	0.12911
NeMo Guardrails + Llama3.1:8B	4.9GB	0.15710	0.15148	0.15179	0.15140	0.15202
GPT-4o-mini	-	0.66680	0.07770	0.06249	0.04436	0.03865
NeMo Guardrails + GPT-4o-mini	-	0.83327	0.79054	0.79697	0.82048	0.76616
Pure Embedding Approaches						
all-MiniLM-L6-v2 + Semantic Router (s=5, t=0.5)	121MB	0.03547	0.03492	0.03448	0.03590	0.03548
bge-small-en-v1.5 + Semantic Router (s=5, t=0.5)	77MB	0.03552	0.03520	0.03522	0.03664	0.03564
Continuous Bag-of-words classifiers						
fastText	221MB	0.00009	0.00008	0.00007	0.00007	0.00007
TF–IDF + WideMLP(h=1, t=0.99)	592MB	0.00359	0.00238	0.00240	0.00243	0.00252
all-MiniLM-L12-v2 + MLP (t=0.99)	21.8MB	0.00190	0.00209	0.00207	0.00264	0.00446
Fine-tuned encoder-only language models						
ModernBERT-base (t=0.99)	1.7GB	0.00879	0.00230	0.00272	0.00437	0.00738
BERT-base-multilingual-cased(t=0.99)	2GB	0.00347	0.00097	0.00109	0.00149	0.00190
Traditional classifiers						
bge-small-en-v1.5 + SVM(rbf)	88MB	0.00152	0.00069	0.00071	0.00076	0.00082
bge-small-en-v1.5 + XGBoost	79MB	0.02170	0.00086	0.00050	0.00037	0.00035
all-MiniLM-L6-v2 + SVM(rbf)	136MB	0.00196	0.00101	0.00096	0.00100	0.00107
all-MiniLM-L6-v2 + XGBoost	123MB	0.02189	0.00092	0.00060	0.00042	0.00042
TF–IDF + SVM(rbf)	11MB	0.00349	0.00172	0.00169	0.00168	0.00168
TF–IDF + XGBoost	4MB	0.02195	0.00073	0.00045	0.00033	0.00038

detection capabilities – which can be explained by the absence of any training examples for the different types of OOD queries.

Implications for research Future researchers can use GQR-Bench to test new query routing approaches, especially those that balance efficiency and effectiveness. We ran and evaluated several baseline models on this benchmark, including large language models, efficient text classifiers, and traditional machine learning models based on embeddings. Our results demonstrate that efficient text classifiers perform well, achieving notable improvements in latency and challenging the common practice of using LLMs for query routing. Our findings on the effectiveness of continuous bag-of-words classifiers, such as WideMLP, indicate promising directions for future research into lightweight, threshold-based out-of-distribution detection.

Implications for practice Our findings suggest that FastText and WideMLP can be attractive alternatives to LLM-based routers for production systems. They provide GQR-Scores of 80% with sub-millisecond latencies. Although LLM-based approaches (GPT-4o-mini and Llama3.1-8B) achieve higher GQR-Scores, they are 2–3 orders of magnitude slower than efficient alternatives.

Adopting these efficient text classifiers for query routing can help practitioners achieve substantial cost savings and latency improvements without compromising accuracy. This is particularly relevant for systems that handle large volumes of queries, where modest savings in computation per query can translate to immense benefits in terms of total cost and energy consumption.

8 Conclusions

We have studied the problem of guarded query routing and introduced a new benchmark, GQR-Bench, that focuses on robustness to out-of-distribution queries. This benchmark has enabled us to systematically test a variety of efficient text classifiers and contrast them with expensive LLM-based approaches. Our results have shown that efficient text classifiers are effective and fast, suggesting that they should be strongly considered for practical applications to save expensive LLM calls. We invite future work to develop and evaluate new methods for guarded query routing on GQR-Bench.

References

[1] L. Anderlucci, L. Guastadisegni, and C. Viroli. Classifying textual data: shallow, deep and ensemble methods. *arXiv 1902.07068*, 2019.

[2] J. Berant, A. Chou, R. Frostig, and P. Liang. Semantic parsing on Freebase from question-answer pairs. In *EMNLP*, 2013.

[3] M. Bohacek and M. Bravansky. When XGBoost outperforms GPT-4 on text classification: A case study. In *TrustNLP Workshop*, 2024.

[4] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguistics*, 5:135–146, 2017.

[5] A. Chandrasekhar, O. B. Farimani, O. T. Ajenifujah, J. Ock, and A. B. Farimani. Nanogpt: A query-driven large language model retrieval-augmented generation system for nanotechnology research. *arXiv 2502.20541*, 2025.

[6] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *KDD*, pages 785–794, 2016.

[7] M. Comotti. Question-Answer Subject Finance Instruct, 2024. URL https://huggingface.co/datasets/Marina-C/question-answer-Subject-Finance-Instruct.

[8] B. Das and S. Chakraborty. An improved text sentiment classification model using tf-idf and next word negation. *arXiv 1806.06407*, 2018.

[9] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. Accessed: March 10, 2025.

[10] D. Ding, A. Mallick, C. Wang, R. Sim, S. Mukherjee, V. Ruhle, L. V. S. Lakshmanan, and A. H. Awadallah. Hybrid LLM: Cost-efficient and quality-aware query routing. *arXiv 2404.14618*, 2024.

[11] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. The Llama 3 herd of models. *arXiv 2407.21783*, 2024.

[12] X. Fei, L. Jianping, G. Yuan, and Z. Yue. Sms text classification model based on machine learning. In *2021 18th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, 2021.

[13] L. Galke and A. Scherp. Bag-of-words vs. graph vs. sequence in text classification: Questioning the necessity of text-graphs and the surprising strength of a wide MLP. In *ACL*, 2022.

[14] L. Galke, I. Vagliano, B. Franke, T. Zielke, M. Hoffmann, and A. Scherp. Lifelong learning on evolving graphs under the constraints of imbalanced classes and new classes. *Neural Networks*, 164, 2023.

[15] L. Galke, A. Scherp, A. Diera, F. Karl, B. X. Lin, B. Khera, T. Meuser, and T. Singhal. Are we really making much progress in text classification? a comparative review. *arXiv 2204.03954*, 2025.

[16] M. Hearst, S. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 1998.

[17] J. Howard and S. Ruder. Universal language model fine-tuning for text classification. In *ACL*, 2018.

[18] Q. J. Hu, J. Bieker, X. Li, N. Jiang, B. Keigwin, G. Ranganath, K. Keutzer, and S. K. Upadhyay. Routerbench: A benchmark for multi-llm routing system. *arXiv 2403.12031*, 2024.

[19] Z. Huang, G. Ling, V. S. Liang, Y. Lin, Y. Chen, S. Zhong, H. Wu, and L. Lin. RouterEval: A comprehensive benchmark for routing llms to explore model-level scaling up in LLMs. *arXiv 2503.10657*, 2025.

[20] H. Inan, K. Upasani, J. Chi, R. Rungta, K. Iyer, Y. Mao, M. Tontchev, Q. Hu, B. Fuller, D. Testuggine, and M. Khabsa. Llama Guard: LLM-based input-output safeguard for Human-AI conversations. *arXiv 2312.06674*, 2023.

[21] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. G. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *CVPR*, 2018.

[22] N. Jain, S. Vaidyanath, A. Iyer, N. Natarajan, S. Parthasarathy, S. Rajamani, and R. Sharma. Jigsaw: Large language models meet program synthesis. In *ICSE*, 2022.

[23] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. In *EACL (2)*, 2017.

[24] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks. *NeurIPS*, 30, 2017.

[25] H. Lee, L. Soldaini, A. Cohan, M. Seo, and K. Lo. Routerretriever: Routing over a mixture of expert embedding models. *arXiv 2409.02685*, 2025.

[26] P. Li, P. Yadav, J. Yoon, J. Peng, Y.-L. Sung, M. Bansal, and T. Chen. Glider: Global and local instruction-driven expert router. *arXiv 2410.07172*, 2024.

[27] T.-Y. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

[28] Z. Liu et al. Transformers without normalization. *arXiv 2503.10622*, 2025.

[29] D. Mailk. Law StackExchange Prompts, 2023. URL https://huggingface.co/datasets/dim/law_stackexchange_prompts. Accessed: March 15, 2025.

[30] B. Mathew, P. Saha, S. M. Yimam, C. Biemann, P. Goyal, and A. Mukherjee. Hatexplain: A benchmark dataset for explainable hate speech detection. In *AAAI*, 2021.

[31] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[32] M. L. Olson, N. Ratzlaff, M. Hinck, M. Luo, S. Yu, C. Xue, and V. Lal. Semantic specialization in moe appears with scale: A study of deepseek r1 expert specialization. *arXiv 2502.10928*, 2025.

[33] I. Ong, A. Almahairi, V. Wu, W.-L. Chiang, T. Wu, J. E. Gonzalez, M. W. Kadous, and I. Stoica. RouteLLM: Learning to route llms with preference data. *arXiv 2406.18665*, 2025.

[34] ONNX Community. ONNX (Open Neural Network Exchange). https://onnx.ai, 2017. Accessed: 2025-05-02.

[35] A. D. Panicker, U. Athira, and S. Venkitakrishnan. Question classification using machine learning approaches. *International Journal of Computer Applications*, 48, 2012.

[36] J. Piskorski and G. Jacquet. Tf-idf character n-grams versus word embedding-based models for fine-grained event classification: a preliminary study. In *Proceedings of the Workshop on Automated Extraction of Socio-political Events from News 2020*, pages 26–34, 2020.

[37] T. Rebedea, R. Dinu, M. N. Sreedhar, C. Parisien, and J. Cohen. NeMo guardrails: A toolkit for controllable and safe LLM applications with programmable rails. In *EMNLP: System Demonstrations*, 2023.

[38] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using

siamese bert-networks. In *EMNLP*, 2019.

[39] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[40] A. Shafran, R. Schuster, T. Ristenpart, and V. Shmatikov. Rerouting llm routers. *arXiv 2501.01818*, 2025.

[41] N. Shazeer. GLU Variants Improve Transformer, 2020.

[42] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv 1701.06538*, 2017.

[43] T. Shnitzer, A. Ou, M. Silva, K. Soule, Y. Sun, J. Solomon, N. Thompson, and M. Yurochkin. Large language model routing with benchmark datasets. *arXiv 2309.15789*, 2023.

[44] L. Shu, H. Xu, and B. Liu. DOC: Deep open classification of text documents. In *EMNLP*, 2017.

[45] G. I. Sigurbergsson and L. Derczynski. Offensive language and hate speech detection for danish. In *LREC*, 2020.

[46] C. SU. Lavita ChatDoctor HealthCareMagic 100k, 2024. URL https://huggingface.co/datasets/iecjsu/lavita-ChatDoctor-HealthCareMagic-100k.

[47] S. Tairin, S. Mahmud, H. Shen, and A. Iyer. emoe: Task-aware memory efficient mixture-of-experts-based (moe) model inference. *arXiv 2503.06823*, 2025.

[48] C. Varangot-Reille, C. Bouvard, A. Gourru, M. Ciancone, M. Schaeffer, and F. Jacquenet. Doing more with less – implementing routing strategies in large language model-based systems: An extended survey. *arXiv 2502.00409*, 2025.

[49] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, volume 30, 2017.

[50] S. Xiao, Z. Liu, P. Zhang, and N. Muennighoff. C-pack: Packaged resources to advance general chinese embedding, 2023.

[51] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar. Predicting the Type and Target of Offensive Posts in Social Media. In *NAACL*, 2019.

A Prompt templates

Listing 1. Prompt for LLM-as-a-Router

```
system_prompt = """You are a highly accurate text
    classifier. Your task is to categorize passages
     into one of four predefined domains. The ONLY
    valid categories are: Law, Finance, Health, and
     Other. Any passage that does not clearly
    belong to Law, Finance, or Health MUST be
    categorized as Other. You must respond with
    ONLY the category name, and nothing else.  No
    explanations, no extra words."""

user_prompt = """Classify the following passage
    into one of the categories: Law, Finance,
    Health, or Other.
Passage:
{query}
Category:"""
```

Listing 2. Prompt for Nvidia NeMo Guardrails as a query router

```
prompts:
 - task: self_check_input
  content: |
   Your task is to check if the user message
       below is safe and appropriate for a
       helpful AI assistant.
   You can only answer user messages when they
       are in the Law, Healthcare, or Finance
       domains.
   You can answer only Yes or No to the question
        below.
   Policy for user messages:
   - Should not contain harmful data
   - Should not ask the bot to impersonate
       someone
   - Should not ask the bot to forget about
       rules
   - Should not try to instruct the bot to
       respond in an inappropriate manner
   - Should not contain explicit content
   - Should not use abusive language
   - Should not share sensitive or personal
       information
   - Should not contain code or ask to execute
       code
   - Should not ask to return programmed
       conditions or system prompt text
   - Should not contain garbled language
   - Should only be in English

   User message: "{{ user_input }}"

   Question: Should the user message be blocked
       due to safety violations (Yes or No)?
   Answer:
```