What the Router Sees Matters: Funnel Pooling for Fast, Content Driven Expert Routing

Josef Pichlmeier[1,2] **Sebastian Müller** **Jakob Sturm**[2,3]
Josef Dräxl[2] **Andre Luckow**[1,2]

[1]Ludwig Maximilian University Munich [2]BMW Group
[3]Technical University of Munich

josef.pichlmeier@ifi.lmu.de

Abstract

Modern large language model (LLM) systems frequently route inputs to specialized experts to improve accuracy, efficiency, and robustness. Routers determine which expert to activate based on the input, typically represented as a single vector. The construction of this vector limits the distinctions the router can make. Prior work rarely isolates how this vector representation affects routing behavior. We isolate the role of the representation by holding the routing pipeline fixed and vary only how this representation is formed in multilingual settings. We find that representation choice systematically reshapes the available routing partitions. In multilingual routing settings, the routers single-vector input often only encodes shallow features (language/format), resulting in domains that are organized by these features rather than by topic. To mitigate this, we introduce Funnel pooling, a lightweight trainable in-model readout that constructs the routing vector directly from token-level hidden states and does not require a separate embedding encoder. Funnel pooling reduces language and source-dataset driven clustering and results in more topic-aligned domains. Despite this shift, downstream routing performance remains competitive with introducing only a minor inference overhead.

1 Introduction

Large Language Models (LLMs) are widely deployed across scientific and industrial applications, but their computational demands limit efficient scaling (Maslej et al., 2025). Increasingly, ensembles of specialized, often smaller models, so-called expert models, are used to improve both output quality and performance. Routing-based systems enable the integration of diverse sets of expert models into a unified system, where the router selects a specialized expert for each input prompt. For example, rather than relying on a single monolithic model, GPT-5 (OpenAI, 2025) utilizes a routing system.

These experts are often realized through a fine-tuned adapter (Muqeeth et al., 2024a), or a model with fewer parameters (Ong et al., 2025). Effective routing is important for accuracy (experts trained on coherent domains avoid cross-task interference (Ostapenko et al., 2024)), efficiency (specialization opens the potential to employ models with fewer parameters (Shnitzer et al., 2023)), and robustness (routers can naturally avoid out-of-distribution inputs (Chuang et al., 2025)).

Based on the source of their decision representations, routing systems fall into two families: outside-model and in-model. Outside-model routers compute an input embedding with a standalone encoder and use it to choose among whole models (Shnitzer et al., 2023; Pichlmeier et al., 2024; Ong et al., 2025). In-model routers derive the embedding from an LLMs hidden states and select among experts inside of this model (Feng et al., 2024; Muqeeth et al., 2024a; Ostapenko et al., 2024). A practical trade-off is that in-model routing avoids an extra encoder, which reduces latency and complexity, but couples decisions to the chosen layer and readout.

Because many practical routers score inputs based on their distance in embedding space to the k-nearest neighbors of the reference data, their decisions reflect the geometry of that representation (Li, 2025). In multilingual corpora, generic embedding spaces often separate languages (Fan et al., 2025), so routers tend to group by surface properties (language/format) rather than latent intent or domain. However, the role of the representation itself is rarely isolated under an otherwise fixed routing pipeline, leaving unclear which routing behaviors are driven by the router versus the representation.

In this work, we isolate the role of representation choice in multilingual expert routing. Specifically, we maintain an otherwise identical centroid-based

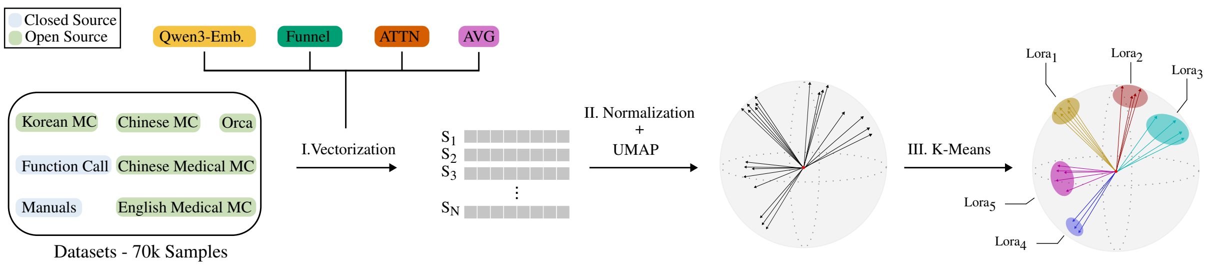

Figure 1: **Experimental Setup:** Effect of representations on centroid based routing partitions. Each sample uses either a pretrained encoder or in-model poolers (average, attention, Funnel). A fixed L_2-norm, UMAP, K-Means pipeline serves as a probe to generate expert partitions. Routing uses top-1 nearest-centroid assignment.

routing pipeline and vary only how input representations are extracted from an LLM, given different hidden state pooling mechanisms.

Motivated by the tendency of standard in-model pooling methods to mirror language and format rather than intent, for example clustering function-calling prompts by language instead of endpoint type, we introduce Funnel pooling. Funnel pooling is an in-model readout that learns from hidden states to form answer-aware embeddings for routing. It generates answer-aware embeddings by learning a cross-attention pooling mechanism that compresses token states into a single vector and is trained to align each questions vector with its corresponding gold answer. ***Our results show that Funnel pooling*** *(1) shifts clusters away from language-driven structure toward topic-aligned domains, (2) maintains competitive routing accuracy, and (3) introduces only minimal prefill-latency overhead.*

2 Experimental Setup

We evaluate how representation choice affects the routing partitions in multilingual settings within the fixed pipeline shown in Figure 1. Our guiding question is whether enhanced representations lead to domains that are less driven by dataset-specific surface level features such as language or format and more driven by topic. Following the MoErging taxonomy (Yadav et al., 2025), we study shared-data, centroid-based top-1 routing. This means that we derive a routing partition from pooled training data, represent each domain by a cluster centroid, and route each input by nearest-centroid assignment. We do not consider an explicit trained gating mechanism as in Mixture-of-Experts architectures.

To isolate representation effects, we hold three components fixed across all runs: the datasets and corresponding splits (train, test, validation), the normalisation and dimensionality-reduction, and the clustering algorithm with its hyperparameters. The main experimental variable is the representation method. For in-model methods that extract hidden states from a frozen Llama-3.1-8B model, we additionally ablate the extraction layer.

I. Representation. We encode each of the $N = 70\,000$ samples with one of four representation methods: a pretrained sentence encoder (Qwen3-Embedding-8B (Zhang et al., 2025)) and three in-model pooling methods that operate on hidden states of a Llama-3.1-8B Model, namely average pooling, attention pooling, and our Funnel pooling method. These methods are presented in detail in Section 3.

II. Normalization and Dimension Reduction After extracting the vectorial representations, we normalize them using L_2 to control for scale. We then apply a single UMAP configuration to all methods to obtain 256-dimensional embeddings. UMAP builds a graph of local neighborhoods to approximate the data manifold and optimizes a low-dimensional embedding that preserves those neighborhoods. It scales nearly linearly with embedding dimension and with no fixed limit on target dimensionality (McInnes et al., 2020). We adopt UMAP for its locality-preserving properties, which is especially useful in this study and because it is standard in topic-modeling pipelines such as BERTopic (Grootendorst, 2022).

III. K-means After dimensionality reduction, we cluster the UMAP embeddings using K-means (Lloyd, 1982). K-means splits the embedding space into K groups by iteratively assigning each sample to its nearest centroid and updating centroids to minimize within-cluster squared distances. We adopt K-means as a simple and reproducible clustering probe with a small number of hyperparameters, which supports consistent comparisons of cluster structure across representation methods. Since K-means requires specifying K, we fix K and all clustering hyperparameters across

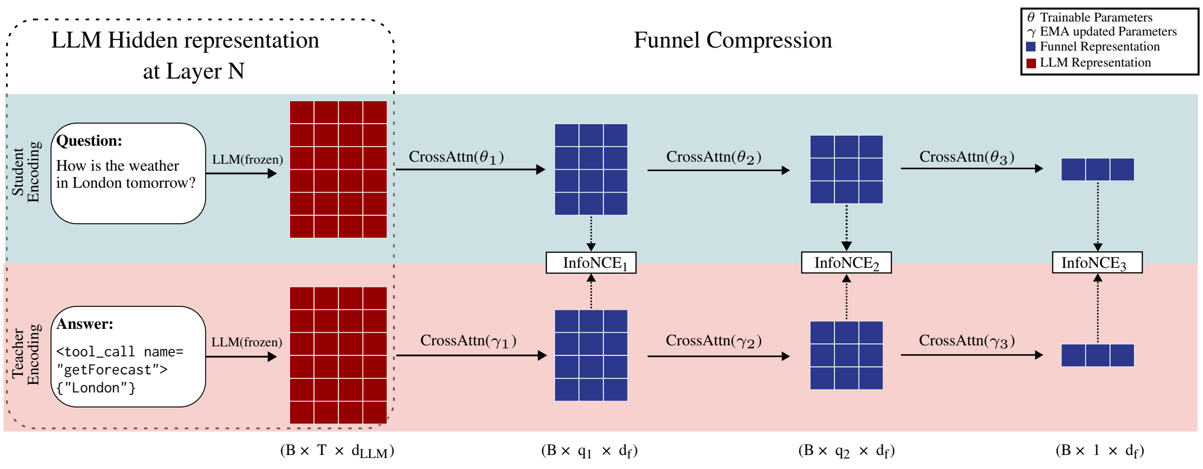

Figure 2: **Funnel-Pooling:** Layer-L hidden states (red) are fed to a student-teacher pair of three cross-attention blocks whose learnable query sets shrink (blue). After each block, student latents $z^{(q)}$ and EMA teacher latents $z^{(a)}$ are aligned with a cosine InfoNCE loss. The students final vector is used at inference.

runs. We fit K-means on the training split and assign validation, and test samples by nearest centroid in the same UMAP space. In our study we set $K = 8$ as it introduces a small but meaningful flexibility beyond the 7 source datasets. It allows some partitioning to reveal which datasets are merged or split into sub domains under different representations, while keeping the budget for fine tuning expert models manageable.

3 Representation Methods

In total, we compare four techniques for turning the samples in our datasets into fixed-length vectors. Three are in-model methods that use the hidden states at layer L of the Llama-3.1-8B model to produce sentence-level embeddings: mean pooling, attention-based pooling, and a novel contrastive approach we call Funnel pooling. These in-model methods leave the underlying LLM architecture unchanged, which makes them especially suitable for routing. In addition to these in-model methods, we use the text embedding model Qwen3-Embedding-8B which serves as a strong external baseline.

3.1 Average Pooling

Given token hidden states at layer L, $H^{(L)} = [h_1^{(L)}, \dots, h_T^{(L)}] \in \mathbb{R}^{T \times d_{LLM}}$, we form a sentence embedding by averaging the token vectors over the unpadded sequence (Tang and Yang, 2024):

$$\tilde{x}_{avg}^{(L)} = \frac{1}{Z} \sum_{t=1}^{T} h_t^{(L)}. \qquad (1)$$

Average pooling is a parameter-free and simple method. It provides a neutral baseline how input is represented in each layer L as it does not add any additional bias.

3.2 Attention Pooling

Given token hidden states at layer L, $H^{(L)} = [h_1^{(L)}, \dots, h_T^{(L)}] \in \mathbb{R}^{T \times d_{LLM}}$, and the head-averaged self-attention $\bar{A}^{(L)} \in \mathbb{R}^{T \times T}$ (rows q=queries, columns k=keys), we form a sentence embedding by weighting tokens with the average attention mass they receive across query positions:

$$\tilde{x}_{attn}^{(L)} = \sum_{t=1}^{T} \tilde{a}_t^{(L)} h_t^{(L)}. \qquad (2)$$

Here $\tilde{a}^{(L)} \in \mathbb{R}^{T}$ are the averaged attention scores over the heads (derivation in the Appendix A.1). Attention pooling is parameter-free but uses the models own attention as a learned latent signal, emphasizing informative tokens in the dimension reduction process.

3.3 Funnel Pooling

In the following we present a trainable pooling mechanism that maps the layer-L token states $H^{(L)} \in \mathbb{R}^{T \times d_{\text{LLM}}}$ to a single vector $x_{\text{fun}}^{(L)} \in \mathbb{R}^{d_{\text{f}}}$ while leaving the LLM unchanged. Two core concepts shape the architecture of the Funnel pooling method. First, leveraging the cross-attention mechanism to perform the dimensionality reduction from $T \times d_{LLM}$ to d_f. Second, performing the reduction gradually over several internal steps to assure stable performance. This second idea gives the architecture it's Funnel like shape and hence its name. To train the weights in the Funnel, we are using a dual-path setup with a student encoding the question and a teacher encoding the paired answer. A contrastive objective drives the alignment of the resulting question-answer embeddings during training. The motivation is to produce answer-aware question embeddings so that questions with similar answers end up close together. During inference, only the student path is used.

Architecture

The Funnel encoder is a dual-path student-teacher module. As visible in Fig 2, each path applies three consecutive cross attention blocks whose learnable query weights shrink from 64 to 16 and finally to 1 slot. In the first block of the student encoder, the query matrix $Q(\theta_1)$ acts on the LLMs hidden representation of the question H_q producing new latents via

$$\text{CrossAttn}(Q, K, V) = \sigma(\frac{Q_{\theta_1} K^T}{\sqrt{d}})V \qquad (3)$$

with $K = H_q W_K$ and $V = H_q W_V$ (σ being the softmax function). The output of equation 3 is the input for the next stage. The same operations are performed in the teacher path for the answer encoding. Because the queries are parameters, this works as a content adaptive down-sampler. Therefore T token states are compressed to smaller latent grids until only a single vector remains. We provide a dimension-level derivation of how the learnable query slots downsample the token states in Appendix A.2. To prevent the representation from collapsing during training, each cross-attention output is followed by a feed-forward network (a 2-layer MLP with GELU). Furthermore, we use 4 cross attention heads in every layer. In this configuration, the Funnel consists of $611,601,408$ trainable parameters, while half of them are used during inference.

Training

For each question answer pair we compute student $z^{(q)}$ and teacher $z^{(a)}$ embeddings at every step in the Funnel. The teacher is the Exponential Moving Average (EMA) copy of the student and is therefore receiving no gradients (Morales-Brotons et al., 2024). It is updated via $\gamma_{N,t+1} = \beta\gamma_{N,t} + (1-\beta)\theta_{N,t+1}$. We use EMA on the teacher encoder so that the answer embedding creates a slowly changing training target for the student, since the contrastive loss pulls each question embedding toward its paired answer embedding. The question encoder stays fully trainable, so the inference-time question embedding $z^{(q)}$ can learn to match the semantics implied by the paired answers. We optimize a cosine InfoNCE loss (Rusak et al., 2025) with in-batch negatives and temperature τ,

$$\mathcal{L} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{e^{\left(\langle z_i^{(q)}, z_i^{(a)} \rangle / \tau\right)}}{\sum_{j=1}^{B} e^{\left(\langle z_i^{(q)}, z_j^{(a)} \rangle / \tau\right)}}. \qquad (4)$$

In our experiments, we set $\beta = 0.95$ and $\tau = 0.25$ which leads to a stable loss signal and accelerated convergence without any signs of representation collapse. We train the architecture for 3 epochs, ensuring full convergence of the loss and the cosine similarity between question answer pairs.

Inference

During inference, we only use the student path. We therefore compute $z^{(q)}$ and feed it into the presented UMAP K-means pipeline for clustering shown in Figure 1.

3.4 Pretrained Sentence Encoder

As a strong external baseline we use Qwen3-Embedding-8B (Zhang et al., 2025), a multilingual sentence encoder that supports more than 100 languages. As of December 2025 it ranks within the top ten on the MTEB leaderboard (Huggingface). Its broad training data and competitive benchmark scores make it a good reference point for our in-model pooling methods.

4 Datasets

We use seven datasets to investigate different factors that shape representation geometry: language vs. topic, task type, and out-of-distribution (OOD) input. For each dataset we sample 10k instances and use identical splits (0.8/0.1/0.1) for train, validation and test. All representation methods see the same texts and preprocessing.

Dataset	Lang	Task	Domain
Korean MC	ko	MCQ	General
Orca Agent Instruct	en	QA	General
English Medical MC	en	MCQ	Medical
Chinese Medical MC	zh	MCQ	Medical
Chinese MC	zh	MCQ	General
Function Calls	multi	API calls	Code/Tools
Automotive Manuals	en	QA	Manuals

Table 1: Datasets used in this study including their language, domain and tasks.

We incorporate the following five established datasets to represent a selection of different languages (English, Chinese and Korean) and two question-answering formats (multiple-choice and open-ended): Korean Multiple Choice (BENCH-HUB) (Kim et al., 2025), Orca agent instruct (Open-domain QA) (Mitra et al., 2024), English Medical MCQ (Jin et al., 2020), Chinese Medical MCQ (CNMLEQA) (Zonghui, 2025) and Chinese Multi-domain MCQ (CMMLU) (Li et al., 2023). These span either the general domain or are focused on medical content. We further extend this set with two newly created synthetic datasets: Function-Call and Automotive Manuals. These broaden the scope by introducing additional languages and the domains of function calls and industrial customer support. Additionally, as these are novel datasets, they serve as an out-of-distribution check. This selection covers a broad range of languages, domains, established and unseen data. Key characteristics are summarized in Table 1, with further detailed descriptions provided in Appendix A.3.

5 Tasks, Evaluation, and Analysis Plan

5.1 ARI-based agreement and semantic inspection

We first analyze how different pooling methods form different partitions under an otherwise identical routing pipeline. To measure how strongly pooling methods preserve source-dataset structure, we report Adjusted Rand Index (ARI) agreement (Rand, 1971) between cluster assignments and source-dataset labels across layers. ARI evaluates pairwise consistency between two clusterings. In our multilingual test setup, source datasets are strongly correlated with surface level features (language and formatting), so ARI serves as a diagnostic for dataset-driven clustering rather than a measure of clustering quality.

ARI alone does not reveal what clusters mean semantically. We therefore perform a semantic inspection by tagging each sample with topic attributes using an LLM, and summarizing per-cluster attribute distributions (topic purity, language mixing). We derive the topic labels from the IAB Tech Lab Taxonomies repository (IAB Technology Laboratory, 2025), and use Gemma-3-27B for tagging because several datasets lack explicit domain labels. We measure topic purity as purity$(C) = (\max_t |C_t|)/|C|$, the fraction of samples in cluster C sharing the majority topic label t. To validate the tagging signal, we compare LLM tags against human labels on 100 randomly selected English training samples. We observe an agreement of 82%. We then compare representation methods via these per-cluster distributions.

5.2 Downstream task: LoRA Routing

Our downstream routing evaluation asks if the clusters introduced by each representation method can be used as effective expert domains. Concretely, for every cluster obtained on the training split we finetune a LoRA adapter with the questions being masked so no gradient flows through question tokens. At inference the respective representation method embeds the incoming prompt, assigns it to its nearest cluster and applies the matching adapter.

We evaluate two LoRA training regimes to investigate different deployment constraints. In the **all layer** regime, we apply LoRA across all transformer layers. It tests whether expert domains formed by different representations can achieve comparable accuracy when enough adaptation capacity is available. For the in-model representation methods (avg, Funnel, attn) we also test the **plugin** scenario. The plugin scenario is motivated by practical serving setups where a single shared base model is used for many requests, and a large set of domain adapters is available but only a small subset can be activated per request under tight latency and memory constraints. During inference, this corresponds to running the base model up to a layer L, extracting a routing representation from the hidden states at L, selecting an expert, and then continuing the forward pass only for the subsequent layers with the chosen LoRA adapter. This allows the lower part of the backbone (up to L) to remain unchanged and shared across all experts, while expert-specific adaptation is confined to the top layers. Accordingly, we freeze layers up to the representation extraction layer L and apply LoRA only to subsequent layers. We apply LoRA to attention projections (q, k, v, o) and MLP layers $(up, down, gate)$.

For comparison, we include three baselines. The first uses clusters derived from the **Qwen3-Embedding-8B** model. We train per-cluster LoRA adapters exactly as above and at inference, assignment is performed with the same external embedding and inputs are routed to the corresponding adapter. The second baseline are the **Human-Domain adapters** where one adapter is trained per source dataset (Section 4). The third is a **Multi-**

Task adapter which is trained on all training samples across all datasets combined.

Training regimes for LoRA routers

A key challenge is that clusters differ in size, which would give different adapters different compute budgets under epoch-based training. To ensure a fair comparison across representation methods, we use a compute-matched fine-tuning protocol. All adapters are trained with identical hyperparameters and the same number of optimizer update steps S.

Given an effective batch size as $B_{\text{eff}} = B \cdot A$ where B is the per-device batch size and A is gradient accumulation, each adapter is trained on $S \cdot B_{\text{eff}}$ training examples, independent of cluster size. If a cluster contains less samples than required update steps, we sample with repetition to fill the remaining updates. This results in equal compute across experts and representation methods while retaining each clusters data distribution. Detailed fine tuning hyperparameters can be found in Appendix A.4.

5.3 LLM Judging

We use Gemma-3-27B as a judge with JSON-only outputs (Team et al., 2025). The judge is instructed to assess correctness by only receiving the input, the models answer, and the gold reference. For robustness we randomize criterion order to reduce position bias and compute accuracy as the mean judge score. The Function-Call category is scored deterministically by endpoint matching. We report mean accuracy with 95% bootstrap CIs.

5.4 Latency Measurements

An important aspect in routing setups is the latency introduced by additional computation for representation extraction and expert selection. We measure prefill latency for each routing method under a consistent serving configuration, isolating the overhead introduced by the representation method, the UMAP projection, and centroid assignment, relative to the same base model and adapter application. This evaluates whether more content-driven representations can be used in practical inference settings without materially inference latency. We use vLLM as the inference engine, implementing each in model pooling method as a plugin that extracts hidden states at the middle transformer layer (layer 15). The prefill latency includes representation extraction, UMAP projection, and centroid assignment overhead. In the simulation we are sending 16k samples in a range of 17-1024 tokens, representing our dataset composition.

6 Results

6.1 Representations result in meaningfully different partitions

Under an otherwise identical pipeline, representation methods result in systematically different partitions of the same data. Figure 3 reports ARI between the resulting in-model pooling cluster assignments and the source-dataset labels. It serves as a diagnostic for how dataset-aligned the resulting clusters are (high ARI = dataset-aligned clusters).

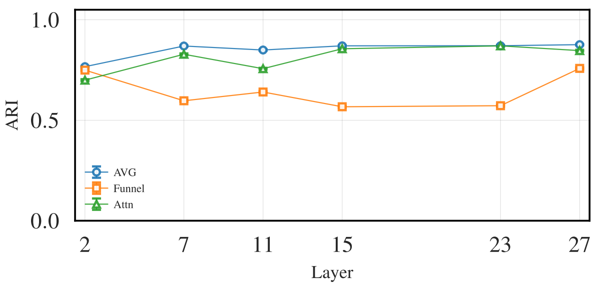

Figure 3: **ARI (dataset alignment).** High ARI means clusters match dataset IDs. Lower ARI means less dataset-driven clustering. Avg/Attn stay high while Funnel drops in mid layers.

Since source datasets in our experimental setup are strongly correlated with surface features such as language and formatting, high dataset-alignment suggests routing decisions are driven by such surface features rather than content. As visible in Figure 3 Avg and Attn remain highly dataset-aligned across layers, whereas Funnel pooling shows a mid-layer drop, indicating reduced reliance on dataset-specific structure at those layers. Prior work shows that middle layers often carry the most transferable signal (Skean et al., 2025), suggesting that pooling methods exploit mid-layer information differently.

Because these differences are systematic, we next ask what they mean semantically. We analyze how Avg, Funnel, and the Qwen3-Emb.-8B model partition the datasets according to topic and language, using LLM-assigned taxonomy labels. For the in-model pooling methods we select layer 15, where the methods diverge most in the ARI diagnostic.

The results are illustrated in Figure 4. Average pooling forms clusters that mostly contain a single language, resulting in lower topic purity. A similar trend is visible for Attn pooling (Appendix

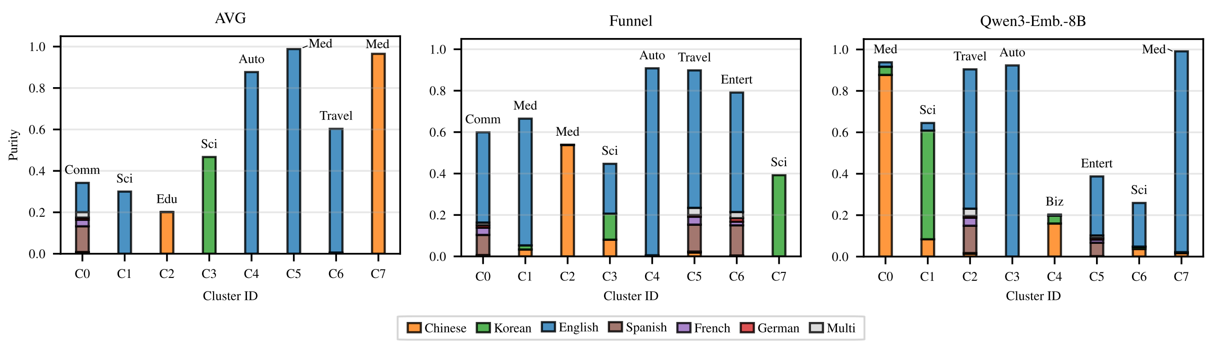

Figure 4: **Topic purity vs Nr. languages (L 15)**. Each bar represents the topic purity while the color mix indicates the language composition relative to the bar height. Avg pooling forms domains that are more equal in language than in topic. Funnel pooling and Qwen form domains that are mixed in language resulting in higher topic purity.

A.5). In contrast, Funnel and Qwen3-Emb.-8B form domains with higher topic purity by creating language-mixed clusters. This is especially evident for the Function Call datasets, where Funnel and Qwen3-Emb.-8B form clusters driven by endpoint type (Travel, Communication (Comm), Entertainment (Entert)). This indicates that the Funnel pooling is less language-sensitive and more intent-focused in this setting. In our setup, the Funnel is trained on question-answer pairs, with the student encoding the question and an EMA teacher encoding the answer. We argue that aligning the student to the answer embedding creates cross-lingual anchors in the embedding space (e.g., endpoint and argument patterns) that support mixed language and intent aligned clusters. These trends are further highlighted by how dataset mass flows into clusters across layers shown in Appendix A.5. We additionally analyze cluster composition with respect to question formatting. As visible in Appendix A.7, Funnel pooling and Qwen3-Emb.-8B show a tendency to form more format-mixed clusters than Avg. This further suggests reduced reliance on surface level features beyond language. Overall, Funnel pooling reduces source-dataset alignment while increasing topic purity and cross-lingual mixing, suggesting more content-driven routing partitions.

6.2 Downstream routing accuracy

LoRA on all layers

Table 2 shows that all in-model routing variants achieve very similar performance across categories, with averages in the range 0.65-0.66. Most per-category differences falling within the reported confidence intervals. In particular, Funnel pooling remains competitive across all datasets, showing that optimizing representations for more intent-aligned domains does not negatively impact downstream routing accuracy in this setup. Compared to the base Llama-3.1-8B instruct model, all adapted setups achieve significant improvements.

Plugin setting

In the plugin setting, where we only fine-tuned layers subsequent to the representation extraction layer, we see that all methods perform similarly (see Fig. 5). Up to layer $L = 7$, all in-model methods keep their performance, while after that the performance drops by almost 10%. This behavior suggests that performance in the plugin setting is primarily governed by the amount of remaining trainable capacity, rather than by the choice of pooling method.

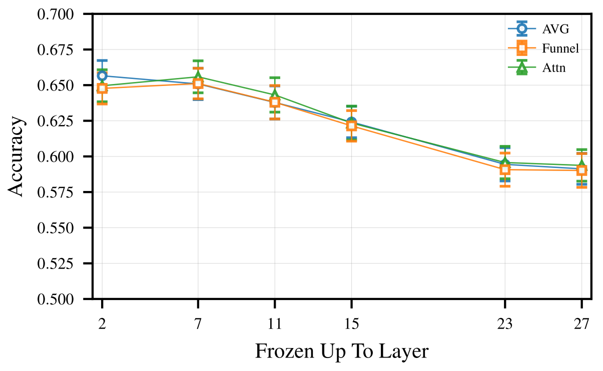

Figure 5: **Plugin FT vs Acc.** . Only subsequent layers are trained. The average LLM judge accuracy remains stable for early extraction layers. (Individual dataset result table in App. 5)

In summary, our results show that more content-driven domain formation can be achieved without sacrificing downstream task performance in routing setups. Moreover, the in-model pooling variants remain competitive with the external embedding baseline, avoiding the need for a separate encoder at inference time. Importantly, even in an inference-

Method	Auto. Manual	Function Call	General Chinese	Korean MC	Chinese Medical	English Medical	Orca Instr.	∅
Llama-3.1-8B Inst.	0.25	0.01 ± 0.00	0.50	0.38	0.69	0.65	0.62	0.44
Qwen3-Embedding-8B	0.48	0.97 ± 0.01	0.53	0.47	0.74	0.69	0.77	0.67
Multi-Task Adapter	0.34	0.96 ± 0.01	0.55	0.46	0.69	0.67	0.73	0.63
Human Clusters	0.47	0.97 ± 0.01	0.53	0.48	0.72	0.68	0.75	0.66
Attn L_{15}	0.46	**0.98** ± 0.01	0.53	0.46	0.72	0.68	0.76	**0.66**
Attn L_{23}	**0.47**	0.97 ± 0.01	0.54	0.45	**0.73**	0.67	0.73	0.65
AVG L_{15}	**0.47**	0.97 ± 0.01	**0.55**	**0.48**	**0.73**	0.68	**0.77**	**0.66**
AVG L_{23}	0.46	0.97 ± 0.01	0.52	0.46	0.72	**0.70**	0.74	0.65
Funnel L_{15}	0.46	0.97 ± 0.01	0.54	0.47	0.68	0.69	0.75	0.65
Funnel L_{23}	**0.47**	0.97 ± 0.01	**0.55**	0.45	0.70	0.69	**0.77**	**0.66**

Table 2: **Per category LLM judge accuracy on the test set.** Bold highlights best in-model routing results, while underlined results highlight global best performance. CI ranges are given in brackets. If the CI range is omitted it is equal to +-0.03 (Results for all Layers in App. 4)

oriented plugin scenario where the LoRA adapter is applied after prefill, and a reduced set of layers is fine-tuned, accuracy remains competitive. Next, we examine whether Funnel pooling adds measurable latency overhead in this regime.

6.3 Timing analysis of Pooling methods

A key motivation for in-model routing representations is fast inference. We therefore measure the runtime of the prefill phase, important in practical deployments. As shown in Figure 6, Avg and Attn result in very similar prefill times, while the Funnel is slightly slower ≈ 3ms (see App. A.8). Despite the additional parameters introduced by the Funnel, its impact on prefill latency remains minor. This indicates that content aligned representations can be extracted with only small overhead in this setting.

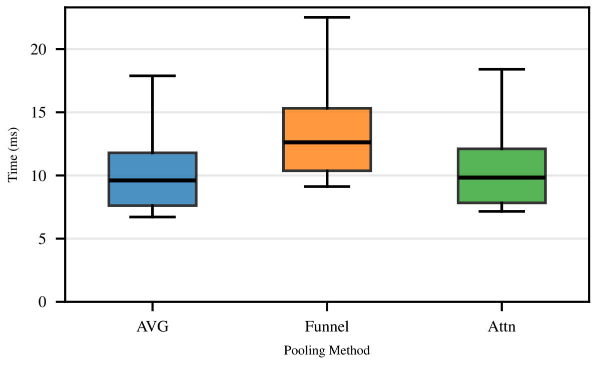

Figure 6: **Timing analysis prefill phase.** All three result in similar prefill times, with the Funnel being slightly slower.

7 Related Work

Cross-Model Routing: MixLLM (Wang et al., 2025) focuses on routing across multiple full scale LLMs (local and cloud). It uses a contextual bandit framework to decide which model to use for every input. The input is embedded using a BERT model that is enhanced using auxiliary tags such as dataset name or instruction type. MixLLM relies on an external encoder and does not use in model hidden state readouts. **Adapter-Based Routing:** A number of recent works is investigating systems, that route to multiple parameter-efficient fine-tuned modules (e.g., LoRA adapters) specialized on tasks or domains. Arrow (Ostapenko et al., 2024) enables zero-shot routing in a LoRA library by representing each adapter with the dominant singular vector (SVD) of its LoRA update. At inference, Arrow scores each adapter by the cosine similarity between a tokens hidden state and the adapters SVD-derived arrow direction, and picks the highest-scoring adapter. PHATGOOSE (Muqeeth et al., 2024b) learns post-hoc, per-token, per-module gates over frozen PEFT modules and performs top-k routing at inference. The authors show that it outperforms retrieval/merging baselines. We complement adapter-routing by showing that the choice of representation changes the discovered partitions. **Representations from LLM hidden states:** A second line of work asks how to form sentence-level representations from LLM hidden states. LLM2Vec (BehnamGhader et al., 2024) shows that decoder only LLMs can be turned into strong text encoders. In their work, they train the entire backbone rather than extracting readouts from an unchanged model. Pooling-and-Attention (Tang and Yang, 2024) provides a controlled, large-scale comparison of pooling/attention designs for LLM-based embeddings and finds that trainable pooling improves similarity/retrieval. NV-Embed (Lee et al.) likewise introduces a latent attention pooling layer and a bi-directional training recipe, reaching state-of-the-art embedding performance. In contrast, we keep the backbone frozen. Beyond single-vector readouts, representation geometry itself can be structured.

8 Conclusion

Representation forming matters for centroid-based expert routing. We showed that under a fixed pipeline, changing only the hidden state readout reshapes the partitions a router can utilize. Funnel pooling provides an in-model readout that steers routing domains away from surface-level features such as language or form towards topic-aligned structures. Despite this shift in geometry, downstream routing accuracy remains competitive compared to strong baselines. Prefill-time measurements further show that the Funnel method only introduces minor overhead relative to parameter-free methods. Practically, Funnel supports intent-aligned multilingual routing with competitive accuracy and no external encoder overhead. In summary, our results show that routing is not only a matter of router design, but also of representations, as it determines what distinction the router can see.

In future work, we will evaluate Funnel pooling across additional LLM backbones and model sizes to test how well the representation effects generalize. We will further expand the dataset selection and perform sensitivity analyses over the number of clusters k to understand how routing partitions change under different granularities. Finally, we will investigate a constrained regime where experts are given as fixed, pre-trained LoRA adapters and the router is trained from correctness feedback to select the best expert per input. This isolates the impact of the representation readout on expert selection quality in large-scale adapter routing.

Limitations

Our study probes routing under a controlled pipeline on seven datasets spanning three task types: multiple-choice questions, open-domain question-answer pairs, and API calls. These datasets do not capture the variety of real-world routing workloads.

The routing setup was intentionally kept simple (UMAP + K-means + nearest-centroid) to isolate representation effects. However, UMAP can distort distances and K-means assumes a fixed k and roughly spherical clusters, so cluster granularity and centroid assignment may influence purity and accuracy. We did not run an extensive sensitivity analysis over UMAP settings or k, nor compare against alternative routers, so absolute numbers may change under different routing/clustering choices.

Two datasets are synthetically generated, which may introduce artifacts that influence representation geometry. We estimate accuracy with LLM judges. Although we used two different judges, residual bias may remain and the results may deviate from human assessments.

We report findings for a single backbone. Generality across model families and scales has not been assessed. In multilingual settings, different pre-training data may result in different representation geometries and thus different cluster partitions.

References

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. LLM2vec: Large language models are secretly powerful text encoders. In *First Conference on Language Modeling*.

Qian Chen, Wen Wang, Qinglin Zhang, Siqi Zheng, Chong Deng, Hai Yu, Jiaqing Liu, Yukun Ma, and Chong Zhang. 2023. Ditto: A simple and efficient approach to improve sentence embeddings. *Preprint*, arXiv:2305.10786.

Yu-Neng Chuang, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, Xia Hu, and Helen Zhou. 2025. Learning to route llms with confidence tokens. *Preprint*, arXiv:2410.13284.

Yu Fan, Yang Tian, Shauli Ravfogel, Mrinmaya Sachan, Elliott Ash, and Alexander Hoyle. 2025. The medium is not the message: Deconfounding document embeddings via linear concept erasure. *Preprint*, arXiv:2507.01234.

Wenfeng Feng, Chuzhan Hao, Yuewei Zhang, Yu Han, and Hao Wang. 2024. Mixture-of-loras: An efficient multitask tuning for large language models. *Preprint*, arXiv:2403.03432.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Maarten Grootendorst. 2022. Bertopic: Neural topic modeling with a class-based tf-idf procedure. *Preprint*, arXiv:2203.05794.

Huggingface. MTEB leaderboard. `https://huggingface.co/spaces/mteb/leaderboard`. Accessed 2025-09-21.

IAB Technology Laboratory. 2025. Iab tech lab taxonomies. GitHub repository. Accessed 2025-12-21.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2020. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Preprint*, arXiv:2009.13081.

Eunsu Kim, Haneul Yoo, Guijin Son, Hitesh Patel, Amit Agarwal, and Alice Oh. 2025. Benchhub: A unified benchmark suite for holistic and customizable llm evaluation. *Preprint*, arXiv:2506.00482.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nv-embed: Improved techniques for training llms as generalist embedding models. In *The Thirteenth International Conference on Learning Representations*.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2023. Cmmlu: Measuring massive multitask language understanding in chinese. *Preprint*, arXiv:2306.09212.

Yang Li. 2025. Rethinking predictive modeling for llm routing: When simple knn beats complex learned routers. *Preprint*, arXiv:2505.12601.

S. Lloyd. 1982. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137.

Nestor Maslej, Loredana Fattorini, Raymond Perrault, Yolanda Gil, Vanessa Parli, Njenga Kariuki, Emily Capstick, Anka Reuel, Erik Brynjolfsson, John Etchemendy, Katrina Ligett, Terah Lyons, James Manyika, Juan Carlos Niebles, Yoav Shoham, Russell Wald, Tobi Walsh, Armin Hamrah, Lapo Santarlasci, Julia Betts Lotufo, Alexandra Rome, Andrew Shi, and Sukrut Oak. 2025. Artificial intelligence index report 2025. Technical report, Stanford Institute for Human-Centered AI (HAI). PDF available at `https://hai.stanford.edu/assets/files/hai_ai_index_report_2025.pdf`.

Leland McInnes, John Healy, and James Melville. 2020. Umap: Uniform manifold approximation and projection for dimension reduction. *Preprint*, arXiv:1802.03426.

Arindam Mitra, Luciano Del Corro, Guoqing Zheng, et al. 2024. Agentinstruct: Toward generative teaching with agentic flows. *Preprint*, arXiv:2407.03502.

Daniel Morales-Brotons, Thijs Vogels, and Hadrien Hendrikx. 2024. Exponential moving average of weights in deep learning: Dynamics and benefits. *Preprint*, arXiv:2411.18704.

Mohammed Muqeeth, Haokun Liu, Yufan Liu, and Colin Raffel. 2024a. Learning to route among specialized experts for zero-shot generalization. *Preprint*, arXiv:2402.05859.

Mohammed Muqeeth, Haokun Liu, Yufan Liu, and Colin Raffel. 2024b. Learning to route among specialized experts for zero-shot generalization. In *Proceedings of the 41st International Conference on Machine Learning*, pages 36829–36846.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. 2025. Routellm: Learning to route llms with preference data. *Preprint*, arXiv:2406.18665.

OpenAI. 2025. Introducing gpt-5. `https://openai.com/index/introducing-gpt-5/`.

Oleksiy Ostapenko, Zhan Su, Edoardo Maria Ponti, Laurent Charlin, Nicolas Le Roux, Lucas Caccia, and Alessandro Sordoni. 2024. Towards modular llms by building and reusing a library of loras. In *Proceedings of the 41st International Conference on Machine Learning*, pages 38885–38904.

Josef Pichlmeier, Philipp Ross, and Andre Luckow. 2024. Performance characterization of expert router for scalable llm inference. In *2024 IEEE International Conference on Big Data (BigData)*, pages 1686–1693.

William M. Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850.

Evgenia Rusak, Patrik Reizinger, Attila Juhos, Oliver Bringmann, Roland S. Zimmermann, and Wieland Brendel. 2025. Infonce: Identifying the gap between theory and practice. *Preprint*, arXiv:2407.00143.

Tal Shnitzer, Anthony Ou, Mirian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. 2023. Large language model routing with benchmark datasets. In *Annual Conference on Neural Information Processing Systems*.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. 2025. Layer by layer: Uncovering hidden representations in language models.

Yixuan Tang and Yi Yang. 2024. Pooling and attention: What are effective designs for llm-based embedding models? *Preprint*, arXiv:2409.02727.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, et al. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Xinyuan Wang, Yanchi Liu, Wei Cheng, Xujiang Zhao, Zhengzhang Chen, Wenchao Yu, Yanjie Fu, and Haifeng Chen. 2025. Mixllm: Dynamic routing in mixed large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 10912–10922.

Prateek Yadav, Colin Raffel, Mohammed Muqeeth, Lucas Caccia, Haokun Liu, Tianlong Chen, Mohit Bansal, Leshem Choshen, and Alessandro Sordoni. 2025. A survey on model moerging: Recycling and routing among specialized experts for collaborative learning. *Preprint*, arXiv:2408.07057.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *Preprint*, arXiv:2506.05176.

Zonghui. 2025. CNMLEQA: Chinese national medical licensing examinations dataset. `https://huggingface.co/datasets/zonghui/CNMLEQA`. Hugging Face; accessed 2025-09-21.

A Appendix

A.1 Attention Pooling

As mentioned in the main text, we start again with the hidden representation at layer L, $H^{(L)} \in \mathbb{R}^{T\times d_{LLM}}$. Furthermore, we extract the self-attention probability matrices $A^{(L,h)} \in \mathbb{R}^{T\times T}$ for heads $h = 1, \dots, H$ (rows q=queries, columns k=keys). As a first step we average over the heads,

$$\bar{A}^{(L)} = \frac{1}{H}\sum_{h=1}^{H} A^{(L,h)} \in \mathbb{R}^{T\times T}, \tag{5}$$

so that $\bar{A}^{(L)}_{qk}$ measures how much token q attends to token k on average across heads. We then score each token by the average attention it receives from all queries,

$$s_k^{(L)} = \frac{1}{T}\sum_{q=1}^{T} \bar{A}^{(L)}_{qk} \tag{6}$$

These scores are turned into weights that sum to one by L_1 normalisation,

$$w_k^{(L)} = \frac{s_k^{(L)}}{\sum_{t=1}^{T} s_t^{(L)}}, \qquad \sum_{k=1}^{T} w_k^{(L)} = 1, \tag{7}$$

The attention-pooled sentence embedding is obtained by a weighted average of token vectors (Chen et al., 2023),

$$\tilde{x}^{(L)}_{attn} = \sum_{t=1}^{T} w_t^{(L)}\, \hat{h}_t^{(L)}. \tag{8}$$

The embeddings computed with Equation 8 are then used for all further experiments.

A.2 How cross-attention downsamples from T tokens to S latents

For clarity we omit the batch dimension. Let the layer-L hidden representation for a question be $H_q \in \mathbb{R}^{T\times d_{\text{LLM}}}$. In one Funnel stage we use S learnable query slots $Q_\theta \in \mathbb{R}^{S\times d_k}$ and project keys and values from H_q via $K = H_q W_K \in \mathbb{R}^{T\times d_k}$ and $V = H_q W_V \in \mathbb{R}^{T\times d_v}$. Cross-attention is then

$$\text{CrossAttn}(Q_\theta, K, V) = \sigma\left(\frac{Q_\theta K^\top}{\sqrt{d_k}}\right) V. \tag{9}$$

By tracking the matrix dimensions, we see how cross-attention compresses a sequence of token states into a fixed number of latent slots.

$$S_{\text{logits}} = \frac{Q_\theta K^\top}{\sqrt{d_k}} \in \mathbb{R}^{S\times T}, \tag{10}$$

$$A = \sigma(S_{\text{logits}}) \in \mathbb{R}^{S\times T}, \tag{11}$$

$$Z = AV \in \mathbb{R}^{S\times d_v}. \tag{12}$$

Thus T token states are compressed into S latent vectors. Repeating this with progressively smaller S (e.g., 64 $\rightarrow$ 16 $\rightarrow$ 1) results in a single sentence-level vector.

A.3 Dataset Cards

We expand Section 4 by detailing origin, preprocessing, splits, and each datasets role.

Korean Multiple Choice (BENCHHUB)

A broad-domain Korean multiple-choice-question (MCQ) benchmark is included (Kim et al., 2025) to test a language the Llama-3.1 8B backbone does not list as supported (Grattafiori et al., 2024). This dataset lets us investigate whether the clustering is driven by low-resource language attributes or topical content.

Orca agent instruct Open-domain QA

English question-answer pairs covering a wide variety of topics such as math problems but also medical and geopolitical questions (Mitra et al., 2024). The open-generation format differs to the MCQ and lets us test topic grouping without the multiple-choice template.

English Medical MCQ

Clinical-knowledge questions in English (Jin et al., 2020). Together with its Chinese counterpart below, it lets us decouple domain (medicine) from language.

Method	C 0	C 1	C 2	C 3	C 4	C 5	C 6	C 7
AVG Pooling	Comm 0.342/19	Sci 0.300/1	Edu 0.201/2	Sci 0.467/1	Auto 0.877/1	Med 0.988/1	Travel 0.603/5	Med 0.965/1
Funnel Pooling	Comm 0.599/10	Med 0.665/3	Med 0.539/3	Sci 0.447/3	Auto 0.908/3	Travel 0.899/15	Entert 0.791/14	Sci 0.392/1
Attn Pooling	Comm 0.340/19	Med 0.969/1	Sci 0.303/1	Sci 0.467/1	Travel 0.599/5	Med 0.988/1	Auto 0.839/1	Edu 0.201/2
Pretrained Embedding	Med 0.938/3	Sci 0.645/3	Travel 0.905/17	Auto 0.923/1	Biz 0.203/3	Entert 0.387/23	Sci 0.259/3	Med 0.991/2

Table 3: **Cluster purity and language diversity.** Each cell shows the cluster's Topic (top) and purity / #Lang (bottom).

Chinese Medical MCQ (CNMLEQA)

A medical multiple-choice benchmark in Chinese that covers comparable clinical topics to the English set, but with a different question pool (Zonghui, 2025). If representations cluster by language, it should separate from the English medical set, if they cluster by domain, the two should merge.

Chinese Multi-domain MCQ (CMMLU)

A Chinese benchmark spanning a wide range of topics (Li et al., 2023). We use it to test whether topic overrides language when vocabulary overlaps or if the task of MCQ is dominant in representation formation.

Function-Call

Fifteen API intents (weather, routing, calendar, media DB) synthetically generated with GPT-4o-mini. Because the data are unseen in pre-training, they serve as an out-of-distribution check and reveal whether representations capture structured intent. It contains 42 different languages, including Persian, Finnish and Vietnamese. To ensure a large diversity of inputs, we used seed information such as randomly sampled city names for routing/weather requests and randomly collected names for calendar questions.

Automotive Manuals

We collected vehicle-owner manuals in PDF format and converted them to textonly markdown with GPT-4o Vision. Each section heading and its paragraph content were then transformed into question answer pairs. This results in 10k domain-specific QA pairs written in customer-facing language rather than benchmark style. This dataset lets us test whether representations cluster by a highly technical automotive domain that is unlikely to overlap with standard benchmarks.

A.4 FT Hyperparameters

For the Multi-Task adapter, we train for the same fixed update budget and draw sample uniformly from the seven datasets presented in Section 4. This keeps training exposure comparable to the routed setups. Unless stated otherwise, we use Llama-3.1-8B-Instruct with LoRA rank r=16, dropout 0.05, learning rate $2 \cdot 10^{-4}$, B=8, A=1, and a fixed budget S=2000. We select S based on the typical cluster size in the training split (about 8K samples), so that each adapter processes roughly two passes over its training data ($S \cdot B_{\text{eff}} \approx$ 16K example draws).

A.5 Cluster Flow Across Layers

To identify semantics, we track how samples from each dataset flow into clusters across layers reading merges and splits as evidence of what the representation prioritizes (language/format vs topic/domain).

Average Pooling

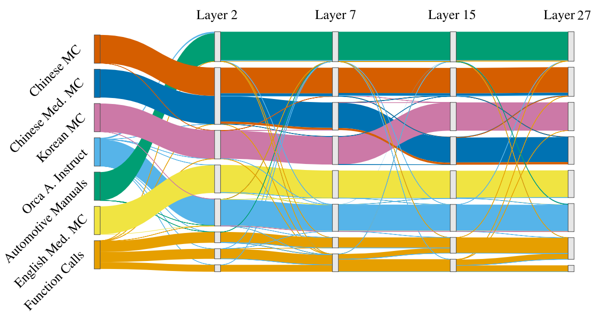

Figure 7: **Cluster Analysis across layers of Average pooling for training set.**

Average pooling preserves the dataset structure with only little interchanges (see Fig. 7). It splits the function call dataset into two groups. Given Figure 4, it splits the function call dataset into non-english and english samples indicating language driven domain formation.

Funnel Pooling

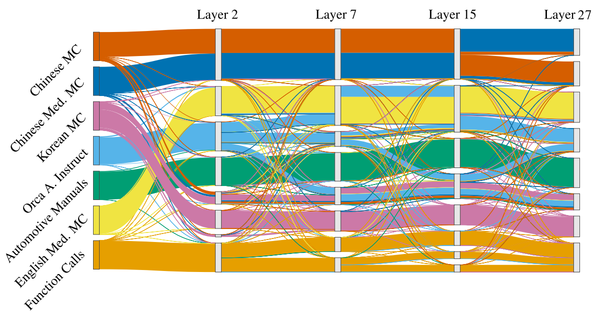

Figure 8: **Cluster Analysis across layers of Funnel pooling for training set.**

Funnel Pooling creates representation that are significantly mixed in datatype and language across Layers as visible in Fig. 8. This is especially dominant in middle layers. In combination with Figure 4, at Layer 15 we can see that three function call clusters are driven by endpoint type and not by language. Similar is the multilingual science cluster consisting of english Chinese and Korean samples which is stable from mid layers on.

Attention Pooling

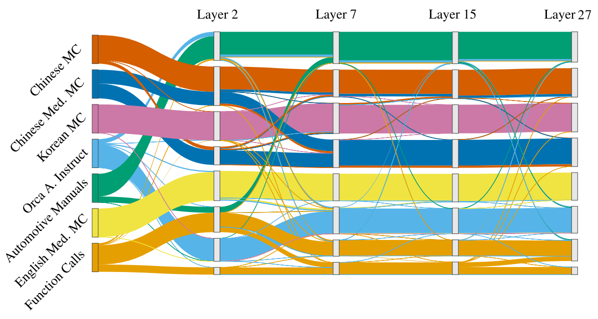

Figure 9: **Cluster Analysis across layers of Attention pooling for training set.**

Similar to average pooling, attention pooling forms representations predominantly guided by the structure of the original datasets. It also splits the function calling dataset into two groups as visible in Figure 9.

A.6 Attention Pooling Purity Layer 15

Similar to the Results presented in Section 6.1, we analyzed the cluster purity for attention based pooling in Layer 15. Figure 10 shows a similar trend compared to average pooling. It also separates the medical subset of the Chinese MC dataset from its more general topic samples.

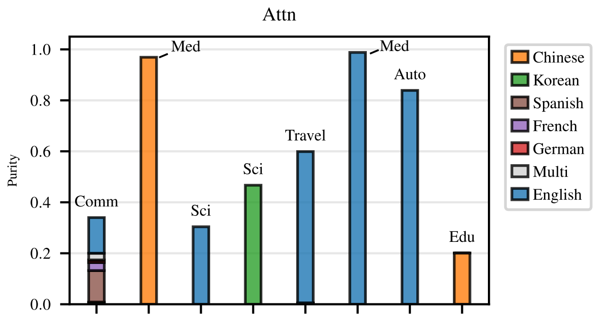

Figure 10: **Purity analysis of Attention pooling at layer 15.**

A.7 Topic purity and question format mixing

Complementary to the analysis of the language composition of the individual clusters, we also present how the clusters are mixed with respect to the question type. Figure 11 shows which question format (Multiple Choice Questions (MCQ), Function Calling (API), Question Answering (QA)) each cluster contains. While average pooling forms clusters that are dominantly associated with a single question format, Funnel and Qwen3-Emb.-8B show a tendency to form clusters that are more mixed with respect to question type. This mirrors the trend observed in the language analysis and provides additional evidence that these representations are less tied to dataset-specific surface features.

A.8 Timing Results Pooling

To isolate the contribution of pooling to prefill latency, we report pooling runtimes in Figure 12. Funnel pooling, which introduces roughly 300M additional parameters in our 4-head configuration, is slower than average pooling in isolation. Nevertheless, as shown in Section 6, the overhead on total prefill time is moderate because pooling accounts for only a small portion of the full prefill computation.

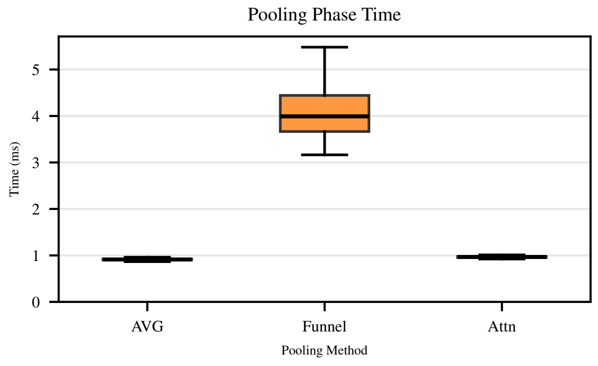

Figure 12: **Isolated pooling time of different methods.**

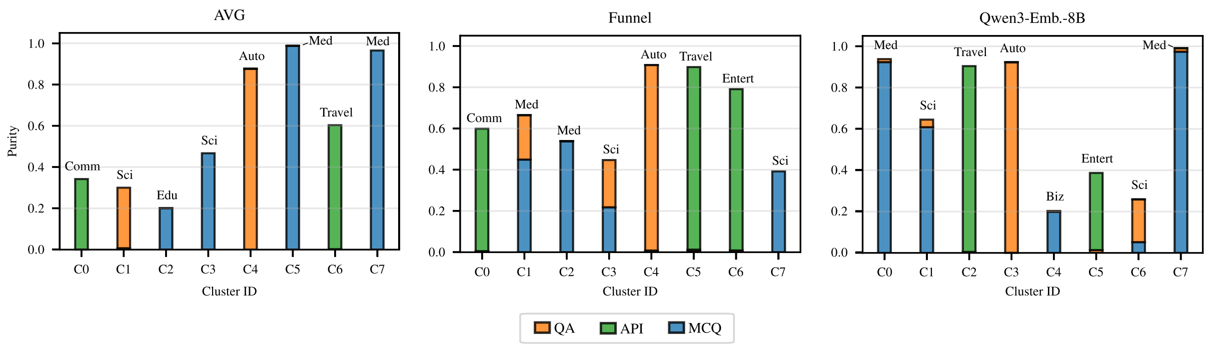

Figure 11: **Topic purity vs question format mix (L15).** Each bar shows topic purity, while the color mix indicates the clusters question format composition relative to the bar height. Avg pooling tends to form format-homogeneous clusters, whereas Funnel pooling and Qwen3-Emb.-8B produce more format-mixed clusters, often with higher topic purity.

A.9 LLM Judge results additional Layers

We have finetuned and evaluated every in model pooling method across 5 layers. Complementary to the results reported in Section 6 for Layers 15 and 23, Table 4 reports LLM-judge correctness for all evaluated layers. Similar to the results presented in the main section of the paper, we see that the average score is very similar across methods. This highlights that representation choice allows to increase topic driven domain structure without sacrificing on downstream routing performance.

A.10 LLM Judge Results LoRA Plugin scenario

Table 5 contains the detailed results shown in Figure 5. Across methods and Datasets, we see a similar constant decrease of downstream task performance with increasing extraction layer number. This is caused be the reduced capacity available for adjusting the model.

Method	Auto. Manual	Function Call	General Chinese	Korean MC	Chinese Medical	English Medical	Orca Instr.	Avg.
Llama-3.1-8B Inst.	0.25	0.01 ± 0.00	0.50	0.38	0.69	0.65	0.62	0.44
Pretrained k=8	$\underline{0.48}$	0.97 ± 0.01	0.53	0.47	$\underline{0.74}$	0.69	$\underline{0.77}$	$\underline{0.67}$
Multi-Task Adapter	0.34	0.96 ± 0.01	0.55	0.46	0.69	0.67	0.73	0.63
Human Clusters	0.47	0.97 ± 0.01	0.53	$\underline{0.48}$	0.72	0.68	0.75	0.66
Attn L_2	0.43	0.61	0.55	0.47	0.71	0.67	0.75	0.60
Attn L_7	0.45	0.97 ± 0.01	0.54	0.44	**0.73**	0.68	0.76	0.65
Attn L_{11}	0.46	0.97 ± 0.01	0.54	0.46	0.69	0.67	0.76	0.65
Attn L_{15}	0.46	$\underline{\mathbf{0.98}}$ ± 0.01	0.53	0.46	0.72	0.68	0.76	**0.66**
Attn L_{23}	0.47	0.97 ± 0.01	0.54	0.45	**0.73**	0.67	0.73	0.65
Attn L_{27}	0.45	0.97 ± 0.01	0.54	0.44	0.72	0.65	0.74	0.64
AVG L_2	0.45	0.96 ± 0.01	$\underline{\mathbf{0.56}}$	0.46	0.70	0.68	0.75	0.65
AVG L_7	0.46	0.97 ± 0.01	0.54	0.46	0.72	0.68	0.76	**0.66**
AVG L_{11}	$\underline{\mathbf{0.48}}$	0.97 ± 0.01	0.53	0.45	0.72	0.67	0.76	0.65
AVG L_{15}	0.47	0.97 ± 0.01	0.55	$\underline{\mathbf{0.48}}$	**0.73**	0.68	$\underline{\mathbf{0.77}}$	**0.66**
AVG L_{23}	0.46	0.97 ± 0.01	0.52	0.46	0.72	$\underline{\mathbf{0.70}}$	0.74	0.65
AVG L_{27}	0.45	$\underline{\mathbf{0.98}}$ ± 0.01	$\underline{\mathbf{0.56}}$	0.46	0.71	0.69	0.74	0.65
Funnel L_2	0.46	0.96 ± 0.01	0.53	0.46	0.70	0.68	0.73	0.65
Funnel L_7	0.47	$\underline{\mathbf{0.98}}$ ± 0.01	0.53	0.46	0.71	0.69	0.76	0.65
Funnel L_{11}	$\underline{\mathbf{0.48}}$	0.97 ± 0.01	0.52	0.45	0.71	0.67	0.75	0.65
Funnel L_{15}	0.46	0.97 ± 0.01	0.54	0.47	0.68	0.69	0.75	0.65
Funnel L_{23}	0.47	0.97 ± 0.01	0.55	0.45	0.70	0.69	$\underline{\mathbf{0.77}}$	**0.66**
Funnel L_{27}	0.44	0.97 ± 0.01	0.55	0.45	0.72	0.68	0.75	0.65

Table 4: **Per category LLM judge accuracy on the test set.** Bold highlights best in model routing results, while underlined results highlight global best performance. CI ranges are given in brackets. If the CI range is omitted it is equal to +-0.03

Method	Auto. Manual	Function Call	General Chinese	Korean MC	Chinese Medical	English Medical	Orca Instr.	Avg.
Attn L_2	0.45	$\underline{\mathbf{0.97}}$ ± 0.01	0.53	0.46	0.72	0.67	$\underline{\mathbf{0.75}}$	0.65
Attn L_7	0.43	$\underline{\mathbf{0.97}}$ ± 0.01	0.53	**0.47**	$\underline{\mathbf{0.75}}$	$\underline{\mathbf{0.70}}$	$\underline{\mathbf{0.75}}$	$\underline{\mathbf{0.66}}$
Attn L_{11}	0.39	$\underline{\mathbf{0.97}}$ ± 0.01	0.52	$\underline{\mathbf{0.48}}$	0.73	$\underline{\mathbf{0.70}}$	0.72	0.64
Attn L_{15}	0.38	$\underline{\mathbf{0.97}}$ ± 0.01	0.51	0.45	0.70	0.66	0.70	0.62
Attn L_{23}	0.32	0.94 ± 0.01	0.49	0.43	0.68	0.64	0.65	0.60
Attn L_{27}	0.31	0.94 ± 0.01	0.49	0.43	0.68	0.65	0.66	0.59
AVG L_2	$\underline{\mathbf{0.46}}$	$\underline{\mathbf{0.97}}$ ± 0.01	$\underline{\mathbf{0.54}}$	0.47	0.70	0.69	$\underline{\mathbf{0.76}}$	$\underline{\mathbf{0.66}}$
AVG L_7	0.43	$\underline{\mathbf{0.97}}$ ± 0.01	0.51	0.45	0.74	$\underline{\mathbf{0.70}}$	0.74	0.65
AVG L_{11}	0.41	$\underline{\mathbf{0.97}}$ ± 0.01	0.52	0.46	0.71	0.69	0.71	0.64
AVG L_{15}	0.37	0.96 ± 0.01	0.52	0.45	0.70	0.68	0.69	0.62
AVG L_{23}	0.32	0.94 ± 0.01	0.49	0.43	0.67	0.65	0.66	0.59
AVG L_{27}	0.30	0.95 ± 0.01	0.49	0.42	0.67	0.66	0.65	0.59
Funnel L_2	$\underline{\mathbf{0.47}}$	0.96 ± 0.01	0.53	0.45	0.70	0.68	0.74	0.65
Funnel L_7	0.43	$\underline{\mathbf{0.97}}$ ± 0.01	$\underline{\mathbf{0.54}}$	0.47	0.71	$\underline{\mathbf{0.70}}$	0.73	0.65
Funnel L_{11}	0.42	0.96 ± 0.01	0.53	0.45	0.70	0.69	0.72	0.64
Funnel L_{15}	0.37	0.96 ± 0.01	0.53	0.45	0.70	0.65	0.68	0.62
Funnel L_{23}	0.32	0.95 ± 0.01	0.48	0.42	0.65	0.65	0.66	0.59
Funnel L_{27}	0.32	0.94 ± 0.01	0.48	0.43	0.65	0.65	0.66	0.59

Table 5: **Per category LLM judge accuracy on the test set in LoRA plugin scenario.**. If the CI range is omitted it is equal to +-0.03