# Understanding Representation Effects in Large Language Model Routing

**Anonymous ACL submission**

## Abstract

Modern large language model (LLM) systems frequently route inputs to specialized experts to improve accuracy, efficiency, and robustness. Routers decide which expert to activate based on the input, which is typically represented as a single vector. The construction of this vector and the space it lives in limits the distinctions the router can make, an effect that prior work rarely examines. We hold the routing pipeline fixed and vary only the representation, showing that this choice reshapes cluster structure and downstream performance. To improve routing, we introduce funnel pooling, which trains the input vector against the answer such that embeddings reflect task intent rather than surface forms. It builds this vector directly from the LLMs token-level hidden states, so no external embedding model is required. Our experiments show that LoRA routing based on funnel-pooling is a competitive choice, outperforming simple token-averaging methods and otherwise matching strong baselines. We observe that funnel pooling forms more language-independent clusters that track task intent instead of surface features. These results show that representation design sets the routers decision boundaries and, in turn, the accuracy and robustness of routed systems across languages.

## 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 as well as performance. Routing-based systems allow the integration of diverse sets of expert models into a unified system, in which 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), a tool (Mu et al., 2024), 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)).

For this work, we categorize routing implementations into two families: outside-model and in-model. Outside-model routers compute an input embedding with a stand-alone encoder and use it to choose among whole models or tools (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.

In both families, routing decisions are primarily made from input representations, so the choice of representation determines the partitions available to the router. 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.

In this work, we show that input representation alone reshapes the partitions a router can exploit and, in turn, affects routing outcomes. Motivated by the tendency of standard in-model pooling methods to mirror language and format rather than in-

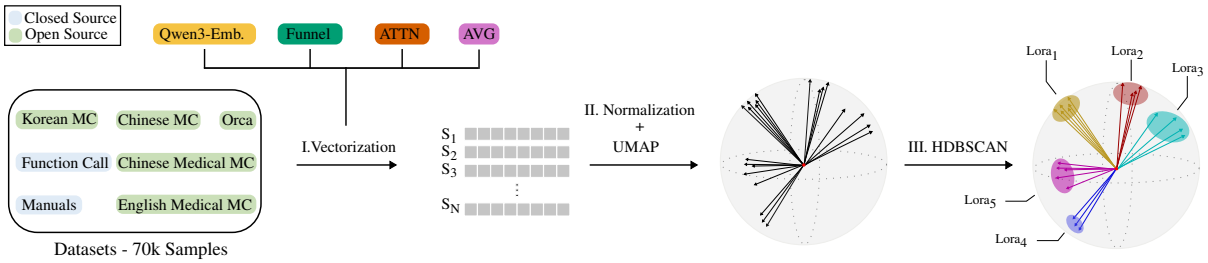


Figure 1: **Experimental Setup:** Effect of representations on clustering and LoRA routing. Each sample uses either a pretrained encoder or in-model poolers (mean, attention, funnel). A fixed $L_2$-norm $\rightarrow$ UMAP $\rightarrow$ HDBSCAN pipeline yields clusters as experts. Data splits are identical.

tent, we introduce funnel pooling, an in-model pooling module 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. In this embedding space, we treat the resulting clusters as expert domains and route to per-cluster LoRA experts, which improves performance relative to established in-model pooling baselines (Table 2). Furthermore, we highlight that funnel pooling tends to group prompts by domain and intent rather than by surface properties, often mixing languages within the same cluster.

We present three main contributions:

- **Representation matters.** Under a fixed experimental setup, different representation methods result in systematically different cluster partitions of the same data.
- **Funnel.** We introduce funnel pooling, a trainable cross-attention readout that downsamples LLM hidden states and, in our setup, produces clusters aligned with domain and intent rather than surface attributes such as language.
- **Routing outcomes.** Funnel-based routing is competitive overall, delivering dataset-dependent gains over average and attention pooling and remaining on par with strong baselines (Table 2).

## 2 Experimental Setup

We evaluate how representation choice shapes cluster structure and LoRA routing performance within the fixed experimental pipeline shown in Figure 1. To isolate the effect of the representations, we are aiming to hold three components fixed across all runs: the data and corresponding splits (train, test, validation, paraphrase), 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 backbone, 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 mean 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 to control for scale: pretrained and in-model embeddings use $L_2$ normalization. We then apply a single UMAP configuration to all methods to obtain 30-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. HDBSCAN** After dimensionality reduction, we cluster in the UMAP space with HDBSCAN (Campello et al., 2015). HDBSCAN is a hierarchical extension of DBSCAN that builds a tree over mutual-reachability distances and extracts stable regions as clusters. It can work with varying cluster densities and shapes and returns a cluster assignment together with a noise label. We adopt HDBSCAN since it does not require pre-specifying the number of clusters $K$. Further-

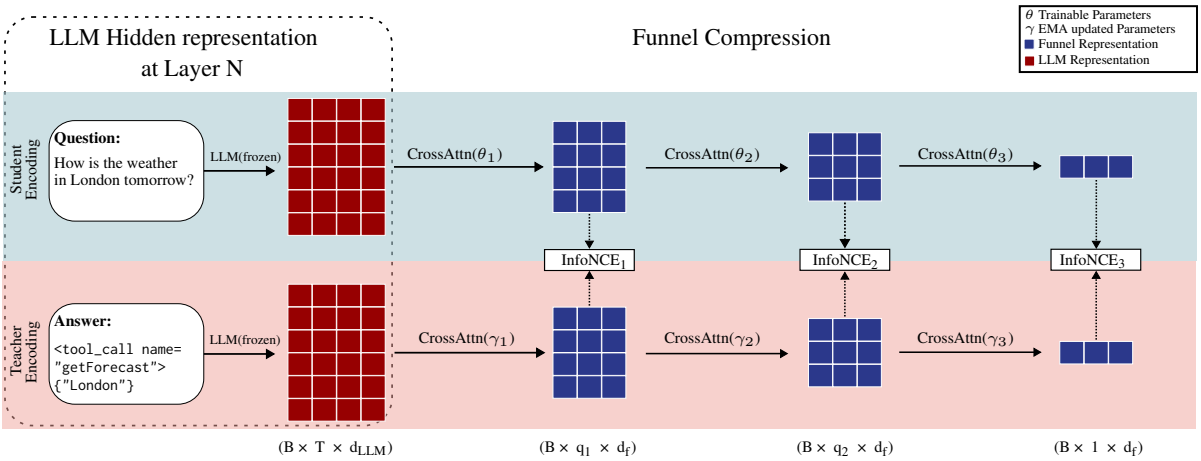


Figure 2: **Funnel-Pooling:** Layer-$L$ hidden states (red) are fed to a student-teacher pair of three cross-attention blocks whose learnable query sets shrink (blue). After each block, student latents $z^{(q)}$ and EMA teacher latents $z^{(a)}$ are aligned with a cosine InfoNCE loss. The students final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

more, the combination of UMAP with HDBSCAN is commonly used in topic discovery (Grootendorst, 2022). UMAP + HDBSCAN cluster in a roughly neighborhood-preserving space, enabling comparison of cluster structure across methods. For each representation method, we fit a UMAP reducer and an HDBSCAN clusterer on the training split, then use the fitted reducer to embed the validation, test, and paraphrase splits and assign clusters. As illustrated in Figure 1, we fine-tune LoRA adapters on the HDBSCAN clusters and then use them for routing. Fine-tuning details are provided in Section 5.1. Detailed hyperparameter settings can be found in Appendix A.6.

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

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

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 $L$, $H^{(L)} = [h_1^{(L)}, \ldots, h_T^{(L)}] \in \mathbb{R}^{T \times d_{LLM}}$, 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:

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

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

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

which are the input for the next stage ($\sigma$ being the softmax function). 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. 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. As visible in Eq. 3, we avoid the usage of $K$ and $V$ projections matrices out of two practical reasons. First to reduce the number of parameters the funnel pooling method would introduce. Second, during early experiments the architecture suffered from dimension collapse which we attributed to implicit regularization (Jing et al., 2022).

**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 EMA copy of the student and is therefore receiving no gradients. It is updated via $\gamma_{N,t+1} = \beta\gamma_{N,t} + (1-\beta)\theta_{N,t+1}$. We optimize a cosine InfoNCE loss (Rusak et al., 2025) with in-batch negatives and temperature $\tau$,

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

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 prestend UMAP HDBSCAN 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 September 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. Furthermore, for all datasets except of the Function Calls dataset, we

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

| 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 |

created a paraphrase set of the training data using GPT-4o.

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 dataset, 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.2.

## 5 Tasks, Evaluation, and Analysis Plan

### 5.1 Downstream task: LoRA Routing

Our downstream evaluation asks if the clusters introduced by each representation method can be used as effective expert domains and if different representations result in different performance. 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. For the in-model representation methods (avg, funnel, attn) we update only layers above the extraction layer $L$, which is the expected plug-in scenario. We apply LoRA to the attention projections $(q, k, v, o)$ and MLP layers $(up, down)$.

For comparison, we include two 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 is a **Multi-Task adapter** which is trained on all training samples across all datasets combined. Prior work reports that multi-task adapters are strong baselines against collections of task-specific adapters (Ostapenko et al., 2024). We therefore use it as a reference to contextualize routing results rather than a target we expect routing to surpass.

#### Training regimes for LoRA routers

We consider two fine-tuning regimes for the in-model routing setups and the Qwen3-Embedding-8B setup. In the **conservative** setting we measure the validation loss and stop training if it fails to improve for eight patience checks. To avoid overfitting tiny clusters we scale the schedule with cluster size. Clusters with less than 500 samples are trained for one epoch at a learning rate of $1 \cdot 10^{-4}$. Clusters between 500 and 1000 samples are trained for two epochs at $1.6 \cdot 10^{-4}$. Larger clusters run for three epochs at $2 \cdot 10^{-4}$. The LoRA rank is fixed at $r = 32$.

To compare the effect of the representation selection without the influence of adjusted training schedules we also apply an **expert-style** fine tuning setting with a fixed training regime. Training details are listed in Appendix A.3.

### 5.2 LLM Judging

We use Gemma-3-27B as a single 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. As a robustness check we replicated scoring with GPT-4o-mini. The results were consistent and did not change any conclusions (Appendix A.7).

### 5.3 ARI-based agreement and semantic inspection

To explain why fine-tuned performance might differ across representations, we introduce two diagnostics, Adjusted Rand Index (ARI) based partition agreement and semantic inspection of clusters (Rand, 1971). ARI evaluates pairwise consistency between two clusterings. For every pair of samples it checks whether both methods place the pair in the same cluster (or both keep it apart) and normalises this count against chance. While ARI tells us how the clusterings align, it does not tell us what the clusters mean. To identify se-

Table 2: **Per category LLM judge correctness 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. |
|---|---|---|---|---|---|---|---|
| LLama-3.1-8B Inst. | 0.25 | 0.01 ± 0.00 | 0.50 | 0.38 | 0.69 | 0.65 | 0.62 |
| Multi-Task Adapter | <u>0.47</u> | <u>0.97</u> ± 0.01 | 0.53 | 0.46 | <u>0.72</u> | <u>0.68</u> | 0.73 |
| Qwen3-Emb.-8B | 0.45 | 0.74 | <u>0.55</u> | 0.46 | 0.66 | 0.64 | 0.68 |
| Attn $L_7$ | 0.38 | 0.91 ± 0.02 | 0.52 | 0.44 | 0.70 | 0.66 | <u>**0.74**</u> |
| Attn $L_{15}$ | 0.35 | 0.88 ± 0.02 | 0.52 | <u>**0.47**</u> | 0.68 | 0.66 | 0.69 |
| AVG $L_7$ | 0.37 | 0.91 ± 0.02 | 0.49 | 0.38 | 0.65 | **0.67** | <u>**0.74**</u> |
| AVG $L_{15}$ | 0.35 | 0.89 ± 0.02 | 0.51 | 0.45 | 0.68 | 0.66 | 0.70 |
| Funnel $L_7$ | **0.43** | **0.96** ± 0.01 | **0.54** | <u>**0.47**</u> ± 0.04 | <u>**0.72**</u> | - | - |
| Funnel $L_{15}$ | 0.39 | 0.95 ± 0.01 | 0.52 | 0.46 | 0.67 | **0.67** | 0.69 |

mantics, we track how samples from each dataset (Sec. 4) flow into clusters across layers (Fig. 3), reading merges and splits as evidence of what the representation prioritizes (language/format vs. topic/domain). For a quantitative view, we tag each sample with attributes $\{\text{topic}, \text{language}, \text{task}\}$ using Gemma-3-27B and summarize per-cluster distributions. We use Gemma-3-27B because some datasets are general-purpose and lack explicit domain labels. We then compare representation methods via these tag distributions, with a focused analysis on the Function-Call dataset.

## 6 Results

For in-model methods we focus on two extraction layers, $L$=7 and $L$=15. This choice is pragmatic because across methods these layers result in strong experts. Furthermore the ARI diagnostics in Sec. 5.3 shows the first cluster differences around $L$=7 which stays stable across mid layers until $L$=15. Together, these layers capture the early-mid layer region where representation effects are most evident as will be explained in Section 6.2. The routing accuracy results for all layers are presented in Appendix A.8.

### 6.1 Routing with per-cluster LoRA experts

Under the conservative regime, funnel pooling delivers small but frequent gains over average and attention pooling. As shown in Table 2, the largest improvements appear on the Function Call and Chinese Medical dataset (absolute deltas up to ∼0.05 - 0.07). For the Auto Manual dataset, the funnel pooling also shows a small but reliable edge (+0.04 - 0.06), whereas General Chinese and Korean QA MC are mixed and typically within ±0.02. For the English Medical at $L$=15 the funnel and average/attention pooling are almost tied (≈ +0.01), and Orca is near parity slightly trailing at $L$=15 (−0.01). Across categories, attention pooling aligns with average pooling, indicating the advantages are specific to funnel pooling and not artifacts of our fixed UMAP-HDBSCAN and fine-tuning setup. The missing funnel entries at $L$=7 for English Medical and Orca occur because, under the fixed HDBSCAN configuration, the $L$=7 cluster algorithm labeled most samples for these categories as noise, resulting in no viable experts for those domains. To check that small test-set clusters do not skew results, we repeat the evaluation on the much larger paraphrase split (48k). Accuracies rise slightly but the method-wise differences mirror the test set, leaving our conclusions unchanged (Appendix A.8, Table 6).

Against the strong pretrained sentence encoder Qwen3-Embedding-8B, the funnel pooling is competitive but result are category dependent (Table 2). It is better on Function Call (0.96/0.95 vs. 0.74) and slightly better on Chinese Medical, Korean MC, English Medical and Orca datasets while it trails on Auto.Manual (0.43/0.39 vs. 0.45) and General Chinese (0.54/0.52 vs. 0.55).

The Multi-Task adapter trained on the full dataset remains a strong reference. Despite each expert seeing less data, funnel-based routing is typically within 0.01-0.04 of the multi-task baseline on five of seven datasets and occasionally exceeds it at $L$=7 (e.g., General Chinese 0.54 vs. 0.53; Korean MC 0.47 vs. 0.46). We use it to contextualize routing results, not as a target we expect routing to exceed in all areas.

To further rule out artifacts from the fine-tuning setup, we also evaluate an expert-style schedule at $L$=15. Detailed results are reported in the Ap-

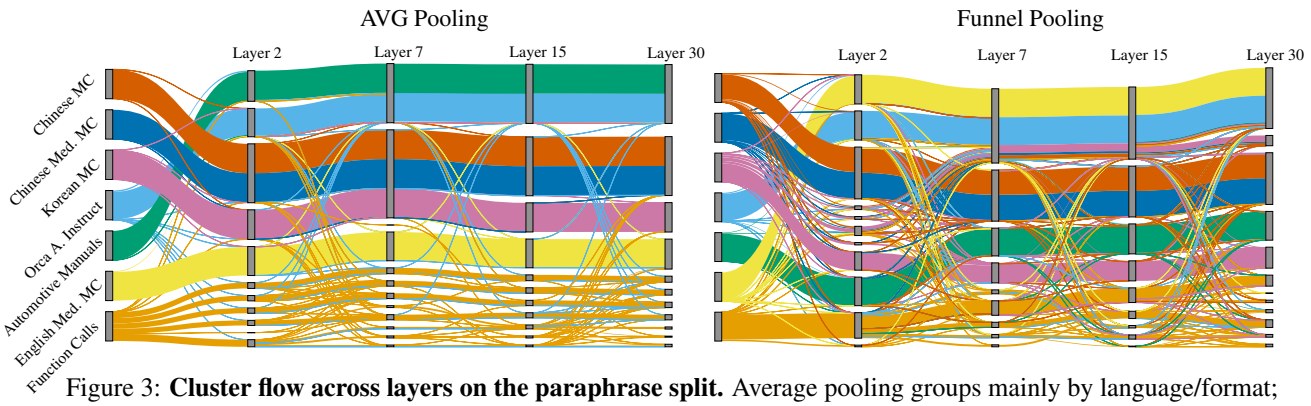


Figure 3: **Cluster flow across layers on the paraphrase split.** Average pooling groups mainly by language/format; funnel pooling groups by domain/intent and mixes languages.

pendix A.3 in Table 3. This fine-tuning regime reveals two distinct patterns. Templatic tasks such as the Function-Call datasets converge for all in-model methods reaching ($\sim 0.96 - 0.97$) while we observe drops on broad/open domains indicating overfitting. These patterns suggest that longer, fixed training leads to mild overfitting on open-ended datasets, and the extra training budget squeezes the performance gaps that come from the choice of representation.

**Takeaway.** Under a conservative, early-stopped schedule at mid-layers, funnel pooling is our most reliable in-model readout, delivering small, frequent gains over average/attention pooling which is largest on Function-Call and Chinese Medical. It remains close to a strong multi-task adapter. With longer fixed training (no early stopping), accuracies converge and open-ended domains dip slightly, compressing representation-driven differences. The above observations motivate using early stopping by default in our experiments, relaxing it only for templatic tasks that saturate.

### 6.2 Representations induce meaningfully different partitions

Because routed accuracy varies across the different representation methods, we next examine the geometry that could explain these differences. Under an otherwise identical pipeline, representation methods result in systematically different partitions of the same datasets. As visible in Figure 4, funnel and average/attention pooling agree at shallow and deep layers but show a mid-layer drop in the ARI score (Details in Appendix A.4). Prior work shows that middle layers often carry the most useful signal (Skean et al., 2025) indicating that the the representation methods utilize this signal differently. Also visible in Figure 4, attention and average pooling maintain high ARI scores across layers with the exception of $L \in \{11, 19, 23\}$. This indicates that the simpler, semi parameter-free methods reveal similar features which forms their cluster structure in similar patterns.

The pretrained encoder partitions the data differently than any in-model pooling method. This results in ARI values for all in-model methods across all layers below ARI $< 0.50$. The exact course can be found in Appendix A.4 Fig. 10.

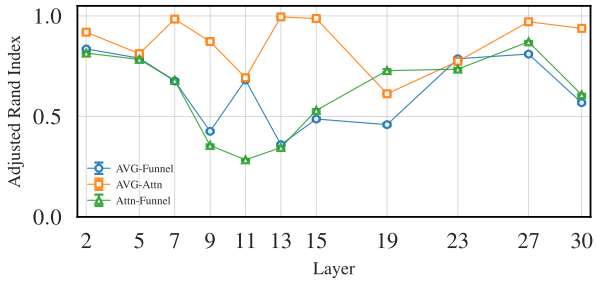


Figure 4: **Pairwise ARI over layers**. The three pairings are indicated by color. Comparing Funnel with other methods gives high ARI in early and later layers. In middle layers a representation induced drop is visible.

Because the differences are systematic and layer dependent, we next ask what they mean semantically. We therefore trace how dataset mass flows into clusters across layers for average and funnel pooling (Fig. 3). The flows suggest that the geometric differences have a semantic direction. Average pooling organizes by language and format, whereas funnel organizes by domain and topic. This is most pronounced in mid layers. Funnel pooling groups technical samples (Medical, Orca, and the engineering subset of the Korean dataset), while average aggregates English QA together. The Qwen3-Embedding-8B model also performs content-driven grouping as explained in Appendix A.5.

As a focused check on semantics, we examine how the Function-Call dataset is partitioned by

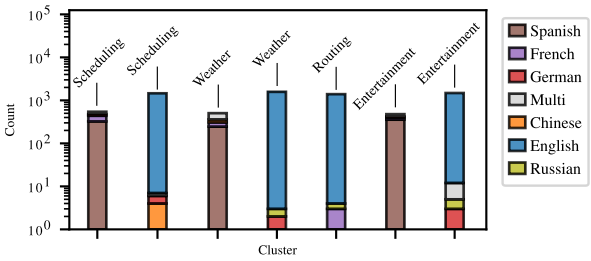


Figure 5: **Cluster Analysis of Average pooling for Function-Call.** Average pooling splits by topic and language, resulting in seven clusters.

topic and language for average versus funnel pooling. Under average pooling, clusters tend to split by function and further split by language (Fig. 5). In contrast, funnel pooling forms language-mixed clusters that align with the function label (Fig. 6). This indicates that the funnel 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. Aligning the student to the answer embedding creates anchors in the embedding space (e.g., endpoint and argument patterns) that are shared across languages.

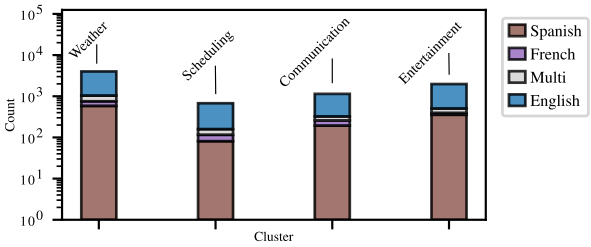


Figure 6: **Cluster Analysis of Funnel pooling for Function-Call.** Funnel separates by topic only, merging languages. Only four clusters emerge.

## 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 and routed outcomes.

**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 routing. By changing only how the input vector is built, the router sees different groups and performance shifts. Funnel pooling works well in practice, improving over simple token averaging and otherwise matching strong baselines. It helps because it learns vectors that track intent or domain rather than surface features, often mixing languages withing a single cluster. However, the effect is sensitive to the training regime. With longer training and no early stopping, method gaps shrink with the risk of overfitting. In general, a controllable trainable representation lets us steer the embedding toward intent and away from surface form (e.g., language), yielding more stable routing decisions.

## 9 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.

Several datasets and paraphrases 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.

Each representation is evaluated under the same procedure, but outcomes are sensitive to HDBSCAN hyperparameters, which mainly affect the number of clusters. We do not provide a full ablation of this sensitivity.

We report findings for a single backbone. Generality across model families and scales has not been assessed. In multilingual settings, different pretraining 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*.

Ricardo J. G. B. Campello, Davoud Moulavi, Arthur Zimek, and Jörg Sander. 2015. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Trans. Knowl. Discov. Data*, 10(1).

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.

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.

Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. 2022. Understanding dimensional collapse in contrastive self-supervised learning. *Preprint*, arXiv:2110.09348.

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.

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.

Feiteng Mu, Yong Jiang, Liwen Zhang, Liuchu Liuchu, Wenjie Li, Pengjun Xie, and Fei Huang. 2024. Query routing for homogeneous tools: An instantiation in the RAG scenario. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10225–10230, Miami, Florida, USA. Association for Computational Linguistics.

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.

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 $h = 1, \ldots, H$ (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),

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

Table 3: The table shows the test accuracy scores for all three in-model methods in the expert-style fine tuning scenario.

| Method$_L$ | Auto. Manual | Function Call | General Chinese | Korean MC | Chinese Medical | Medical MC | Orca Instr. |
|---|---|---|---|---|---|---|---|
| Attn $L_{15}$ | 0.39 | 0.96 | 0.48 | 0.46 | 0.69 | 0.66 | 0.69 |
| AVG $L_{15}$ | 0.38 | 0.97 | 0.49 | 0.45 | 0.69 | 0.65 | 0.68 |
| Funnel $L_{15}$ | 0.41 | 0.97 | 0.49 | 0.44 | 0.68 | 0.67 | 0.68 |

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

### A.2 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.

#### 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.3 Expert-Style fine tuning

To further rule out artifacts from the fine-tuning setup, we also evaluate an expert-style schedule at $L$=15. In this setting, we use a fixed schedule fine tuning regime with hard coded epochs and learning rate. This reduces the influence of early-stopping effects and provides an upper limit on cluster based expert performance. However, it contains the risk of overfitting to smaller cluster, degrading performance on unseen samples. In this setting we train for three epochs with LoRA rank $r = 16$ and learning rate $2 \cdot 10^{-4}$, with early stopping disabled. Results are reported in Table 3.

This fine-tuning regime reveals that accuracies largely converge and we observe drops on

broad/open domains: General Chinese declines for all in-model methods (attn $0.52 \rightarrow 0.48$, avg $0.51 \rightarrow 0.49$, funnel $0.52 \rightarrow 0.49$), Korean MC (funnel $0.46 \rightarrow 0.44$; attn $0.47 \rightarrow 0.46$), and Orca Instr. decreases slightly (avg $0.70 \rightarrow 0.68$; funnel $0.69 \rightarrow 0.68$). In contrast, templatic tasks rise to a shared ceiling on Function-Call (all 0.960.97), and Auto. Manual nudges upward (e.g., funnel $0.39 \rightarrow 0.41$). These patterns suggest that longer, fixed training leads to mild overfitting on open-ended datasets, and the extra training budget squeezes the performance gaps that come from the choice of representation.

### A.4 Detailed ARI Analysis

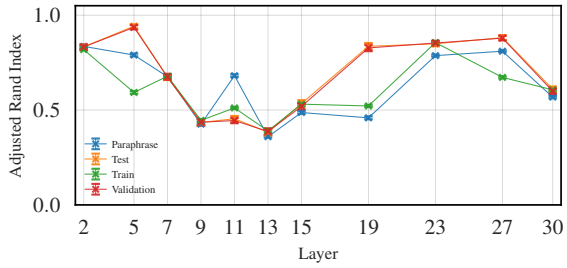


Figure 7: **Average-pool vs. funnel ARI.**. Early/late layers align, mid layers diverge.

For the funnel and avg pooling method, we see the trend, already described in the main text across layers and dataset splits. In early and later layers, a ARI value of $> 0.70$ is indicating a more stable agreement between the clusters which result from both methods. For mid layers we see a divergence between the cluster structures (Fig. 7).

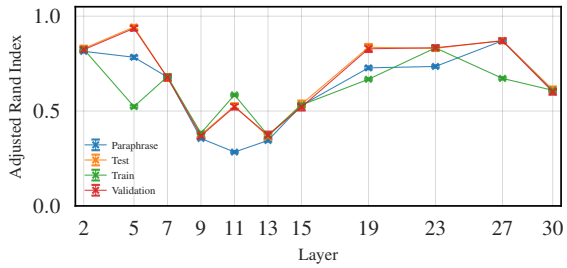


Figure 8: **Attention-pool vs. funnel ARI.**. Early/late layers align, mid layers diverge.

A very similar trend is visible for the ARI trend between attention and funnel pooling. Early and late layers show agreement across splits and middle layers show a more distinct difference. (Fig. 8).

A different trend is visible between average and attention pooling. An almost consistent high ARI value across layers. Minor divergence especially at Layer $L = 11$ is visible (Fig. 9). This also further explains the similar routing scores the two methods show in the main section (Table 2).

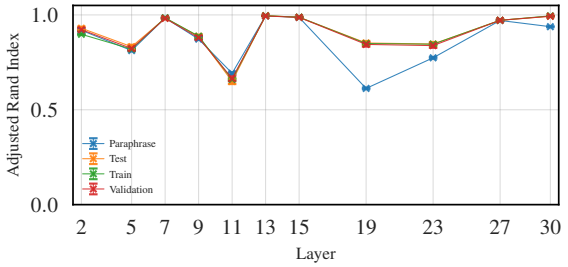


Figure 9: **Average-pool vs. Attention ARI**. High agreement across dataset splits and layers.

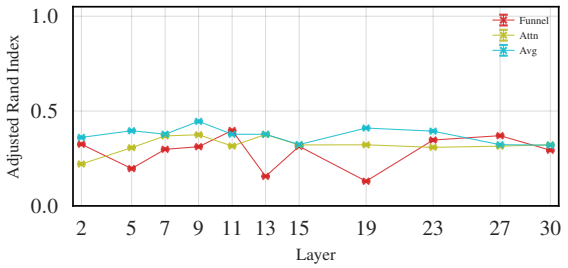


Figure 10: **Pretrained (Qwen-8B-Embedding) vs in-model methods.**. No agreement across methods or layers for paraphrase dataset

None of the in-model methods show similar cluster structures in comparison to the Qwen3-Embedding-8B model as visible in Figure 10. This difference will be further highlighted by the results in Appendix A.5, indicating how this embedding groups the different datasets.

### A.5 Qwen3-Embedding-8B semantic analysis

We examine the pretrained encoder. Fig. 11 shows the dataset co-grouping matrix. Thereby, one cell shows the probability that two samples from the two datasets land in the same HDBSCAN cluster. We observe content-driven grouping with languages mixed. Medical sets co-locate as well as general datasets (Korean MC, Chinese MC, Orca). The Function-Call set is distributed across several intent clusters, which lowers its diagonal self co-grouping.

### A.6 HDBSCAN Hyperparameters

Two HDBSCAN hyperparameters directly affect cluster construction. First, *min_cluster_size* sets

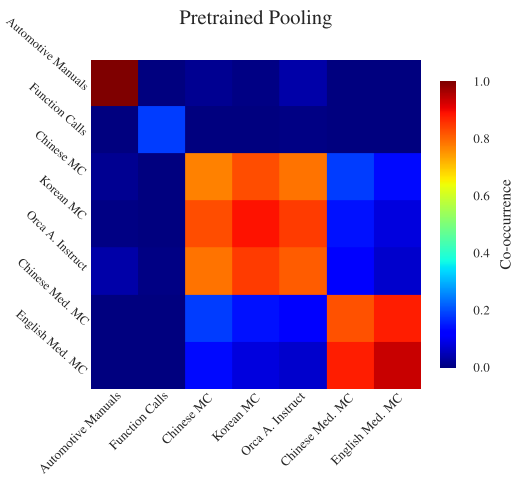


Figure 11: **Dataset Grouping Qwen3-Embedding-8B**. Embedding model groups data by context and not by language.

the minimum group size that qualifies as a cluster.[1] To ensure that each function-call intent (e.g., `getWeather`) can form its own cluster, we set *min_cluster_size* = 450 for all in-model representations. For the pretrained embedding baseline, this setting labeled many function-call samples as noise, so we reduced *min_cluster_size* = 200. This exception resulted in a comparable number in cluster counts relative to the in-model methods (see Table 4, bracketed counts next to method). Second, we set *cluster_selection_epsilon* = 0.3 for all methods. All other parameters used library defaults and were held constant across runs.

### A.7 LLM Judge Results (GPT-4o-mini)

To assess robustness to the choice of judge, we replicated the evaluation with GPT-4o-mini using the same instructions and JSON only format as in the main paper. We report Gemma-3-27B as the primary judge to avoid potential same-family effects, since parts of the data were generated with GPT-4o. Table 5 compares accuracies judged by GPT-4o-mini and Gemma-3-27B. Across datasets and methods, results are very close and no conclusions change.

### A.8 Detailed LLM Judge results Paraphrase dataset

To reduce the influence of small test sets on the results, we also ran a full evaluation on the paraphrase splits using Gemma-3-27B as the judge. The paraphrases were generated with GPT-4o. We did not create paraphrases for the Function-Call dataset because requests are templatic and similar to existing forms. As visible in Table 6 across datasets, most models show higher accuracy than on the original test sets, but the differences between representation methods are unchanged. As a result, our conclusions remain the same.

[1] Documentation: `https://hdbscan.readthedocs.io`

Table 4: Cluster composition by dataset (in-model methods at layer 15). The number in parentheses after each method is the *global* count of HDBSCAN clusters fitted for that method. Each cell reports how many clusters contain samples from the dataset and, in parentheses, the per-cluster training counts. Samples labeled as noise by HDBSCAN are not listed, so the counts in a cell may sum to fewer than 8,000. *Pretrained* uses the external encoder; the other rows are in-model readouts.

| Method | Function Call | Auto. Manual | Chinese Medical | English Medical | Orca | Chinese Multi-dom. | Korean MC |
|---|---|---|---|---|---|---|---|
| Funnel Pooling(8) | 4<br>4000 1979<br>1143 675 | 1<br>7931 | 2<br>7246 751 | 1<br>7904 | 2<br>7703 210 | 2<br>6716 1184 | 2<br>5503 2368 |
| Attention Pooling(10) | 6<br>1464 1379<br>1577 464<br>1456 543 | 1<br>7993 | 1<br>7999 | 1<br>7987 | 1<br>7845 | 1<br>7979 | 1<br>7929 |
| Average Pooling(11) | 7<br>1482 1393<br>1578 1460<br>543 477<br>503 | 1<br>7998 | 1<br>7998 | 2<br>7816 182 | 1<br>7942 | 1<br>7983 | 1<br>7918 |
| Pretrained(12) | 9<br>1975 319<br>1257 194<br>678 376<br>239 1328<br>215 | 1<br>7996 | 2<br>774 7169 | 2<br>265 7705 | 4<br>6910 321<br>276 130 | 3<br>6930 883<br>128 | 2<br>7502 436 |

Table 5: **Per-category LLM judge correctness on the test set (GPT-4o-mini).** Bold highlights the best *in-model routing* result (Attn/AVG/Funnel), while underlining marks the global best across all methods. Values are point estimates with 95% CIs.

| Method | Auto. Manual | Function Call | General Chinese | Korean MC | Chinese Medical | English Medical | Orca Instr. |
|---|---|---|---|---|---|---|---|
| LLama-3.1-8B Inst. | 0.33 ± 0.03 | 0.11 ± 0.02 | 0.49 ± 0.03 | 0.37 ± 0.03 | 0.69 ± 0.03 | 0.64 ± 0.03 | 0.53 ± 0.03 |
| Multi-Task Adapter | 0.46 ± 0.03 | 0.97 ± 0.01 | 0.52 ± 0.03 | 0.44 ± 0.03 | 0.71 ± 0.03 | 0.66 ± 0.03 | 0.67 ± 0.03 |
| Qwen3-Emb.-8B | 0.45 ± 0.03 | 0.74 ± 0.03 | 0.55 ± 0.03 | 0.46 ± 0.03 | 0.66 ± 0.03 | 0.64 ± 0.03 | 0.68 ± 0.03 |
| Attn $L_7$ | 0.37 ± 0.03 | 0.92 ± 0.02 | 0.51 ± 0.03 | 0.43 ± 0.03 | 0.70 ± 0.03 | **0.64** ± 0.03 | **0.68** ± 0.03 |
| Attn $L_{15}$ | 0.33 ± 0.03 | 0.89 ± 0.02 | 0.51 ± 0.03 | **0.46** ± 0.03 | 0.67 ± 0.03 | **0.64** ± 0.03 | 0.63 ± 0.03 |
| AVG $L_7$ | 0.35 ± 0.03 | 0.91 ± 0.02 | 0.48 ± 0.03 | 0.36 ± 0.03 | 0.64 ± 0.03 | **0.64** ± 0.03 | 0.67 ± 0.03 |
| AVG $L_{15}$ | 0.33 ± 0.03 | 0.91 ± 0.02 | 0.50 ± 0.03 | 0.44 ± 0.03 | 0.67 ± 0.03 | **0.64** ± 0.03 | 0.64 ± 0.03 |
| Funnel $L_7$ | **0.41** ± 0.03 | **0.97** ± 0.01 | **0.53** ± 0.03 | **0.46** ± 0.04 | **0.71** ± 0.03 | - | - |
| Funnel $L_{15}$ | 0.37 ± 0.03 | 0.95 ± 0.01 | 0.51 ± 0.03 | 0.45 ± 0.03 | 0.65 ± 0.03 | **0.64** ± 0.03 | 0.63 ± 0.03 |

Table 6: **Per-category LLM judge correctness on the paraphrase set (Gemma-3-27B).** Bold highlights the best *in-model routing* result (Attn/AVG/Funnel), while underlining marks the global best across all methods. Values are point estimates with 95% CIs.

| Method | Auto. Manual | General Chinese | Korean MC | Chinese Medical | English Medical | Orca Instr. |
|---|---|---|---|---|---|---|
| LLama-3.1-8B Inst. | 0.23 ± 0.01 | 0.50 ± 0.01 | 0.38 ± 0.01 | 0.68 ± 0.01 | 0.65 ± 0.01 | 0.61 ± 0.01 |
| Multi-Task Adapter | 0.56 ± 0.01 | 0.87 ± 0.01 | 0.78 ± 0.01 | 0.95 ± 0.00 | 0.96 ± 0.00 | 0.82 ± 0.01 |
| Qwen3-Emb.-8B | 0.59 ± 0.01 | 0.69 ± 0.01 | 0.57 ± 0.01 | 0.72 ± 0.01 | 0.72 ± 0.01 | 0.76 ± 0.01 |
| Attn $L_7$ | 0.39 ± 0.01 | 0.58 ± 0.01 | 0.48 ± 0.01 | 0.72 ± 0.01 | 0.79 ± 0.01 | **0.75** ± 0.01 |
| Attn $L_{11}$ | 0.07 ± 0.01 | 0.51 ± 0.01 | - | 0.72 ± 0.01 | **0.85** ± 0.01 | - |
| Attn $L_{15}$ | 0.39 ± 0.01 | 0.57 ± 0.01 | 0.51 ± 0.01 | 0.73 ± 0.01 | 0.66 ± 0.01 | 0.72 ± 0.01 |
| Attn $L_{23}$ | 0.32 ± 0.01 | 0.51 ± 0.01 | 0.45 ± 0.01 | 0.64 ± 0.01 | 0.63 ± 0.01 | 0.68 ± 0.01 |
| AVG $L_7$ | 0.37 ± 0.01 | 0.48 ± 0.01 | 0.30 ± 0.01 | 0.54 ± 0.01 | 0.76 ± 0.01 | 0.74 ± 0.01 |
| AVG $L_{11}$ | 0.36 ± 0.01 | 0.49 ± 0.01 | 0.42 ± 0.01 | 0.65 ± 0.01 | 0.81 ± 0.01 | 0.71 ± 0.01 |
| AVG $L_{15}$ | 0.37 ± 0.01 | 0.52 ± 0.01 | 0.50 ± 0.01 | 0.65 ± 0.01 | 0.70 ± 0.01 | 0.72 ± 0.01 |
| AVG $L_{23}$ | 0.35 ± 0.01 | 0.48 ± 0.01 | 0.42 ± 0.01 | 0.58 ± 0.01 | 0.63 ± 0.01 | 0.66 ± 0.01 |
| Funnel $L_7$ | **0.54** ± 0.01 | **0.66** ± 0.01 | **0.61** ± 0.01 | **0.80** ± 0.01 | - | - |
| Funnel $L_{11}$ | 0.26 ± 0.01 | 0.52 ± 0.01 | 0.42 ± 0.01 | 0.70 ± 0.01 | 0.63 ± 0.02 | 0.68 ± 0.01 |
| Funnel $L_{15}$ | 0.46 ± 0.01 | 0.51 ± 0.01 | 0.49 ± 0.01 | 0.66 ± 0.01 | 0.65 ± 0.01 | 0.68 ± 0.01 |
| Funnel $L_{23}$ | 0.35 ± 0.01 | 0.51 ± 0.01 | 0.46 ± 0.01 | 0.64 ± 0.01 | 0.63 ± 0.01 | 0.67 ± 0.04 |