# Learning Interaction-Centric Capability Representation for LLM Tool Calling

**Anonymous ACL submission**

## Abstract

As large language models (LLMs) increasingly serve as the backbone of agentic systems with tool-use capabilities, predicting correctness, latency, and cost for a given tool-calling query is essential for efficient model evaluation, selection, and routing. Existing approaches predominantly adopt a semantics-first perspective, without capturing the complex interaction patterns inherent in tool-calling queries and without jointly accounting for cost and latency. We observe that query-model interaction matrices in tool-calling scenarios exhibit a low-rank structure, indicating that behavioral outcomes are governed by latent factors beyond surface semantics. We leverage this observation to propose a dual-pathway framework that decomposes interaction matrices into low-dimensional latent spaces for multi-objective prediction. The **Capability-First (CF)** pathway discovers latent structure via matrix factorization and then projects natural language queries into latent space. The **Joint Capability-Semantic (JCS)** pathway concurrently optimizes the text encoder and model embeddings, implicitly shaping query embeddings through capability-semantic interactions. Experiments on 75 LLMs demonstrate strong performance in predicting correctness, latency, and cost. Further experiments show that our method enables few-shot unseen-model generalization, parameter-efficient training, and interpretable latent representations.

## 1 Introduction

As large language models (LLMs) increasingly serve as the backbone of agentic systems with tool-use capabilities, predicting correctness, latency, and cost for a given tool-calling query is crucial for efficient model evaluation, selection, and routing.

The growing diversity of LLMs, including dense and mixture-of-experts architectures, parameter scales from sub-billion to hundreds of billions, and varying function-calling support, poses a practical challenge: given a query, *which model could handle it?* LLM routing addresses this problem by dynamically selecting the optimal model for each query based on predicted performance (Ong et al., 2024; Chen et al., 2024; Hu et al., 2024; Huang et al., 2025). Existing routing methods predominantly adopt a semantics-first perspective: encoding the query and model representations separately and then predicting the outcome (Jin et al., 2025; Feng et al., 2025; Tsiourvas et al., 2025). However, these approaches present several issues in the tool-calling setting. First, they prioritize query semantics over interaction structure, without revealing how queries and models interact behaviorally. Second, they do not consider the intrinsic complexity of tool-calling queries: despite the binary outcome (success or failure), the underlying reasons involve schema understanding, argument extraction, and tool selection, which are not simply captured by surface semantics alone. Prior studies have shown that tool-calling capability is not monotonically correlated with model size and that function-calling support induces substantial performance differences (Patil et al., 2025; Lin et al., 2025). Third, existing methods lack joint optimization of correctness, latency, and cost, which are all critical in real-world deployment (Ding, 2024; Mu et al., 2024).

These observations suggest that query-model interaction is governed not only by surface semantics, but also by latent structure that shapes how queries and models match. Empirically, we find that the singular value spectrum of three query-model interaction matrices—correctness, latency, and token consumption—constructed from 75 LLMs across 2,000 queries exhibits rapid spectral decay, with effective rank far below the full dimensionality (see Figure 4 in Appendix C.1 for details). This low-rank structure is not an artifact of data sparsity (the matrices are fully observed), but rather reflects

systematic behavioral patterns: query-model interactions admit consistent low-dimensional organization.

This observation suggests that LLM capability prediction can benefit from being reframed as a latent interaction modeling problem. If interactions reside in a low-dimensional space, a natural modeling paradigm is matrix factorization, a well-established technique in collaborative filtering. Just as a user-movie rating matrix can be decomposed into latent preference and feature vectors in collaborative filtering (Koren et al., 2009), a query-model interaction matrix can be decomposed into latent vectors that encode query and model factors.

In this work, we propose a dual-pathway framework for latent capability modeling with two instantiations. **Capability-First (CF)** adopts a two-stage design that separates capability discovery from semantic projection. The Discovery Stage performs matrix factorization on interaction data exclusively, learning query and model embeddings that capture behavioral structure without relying on language features. The Projection Stage trains a language model to project natural language queries into this pre-discovered latent space. **Joint Capability-Semantic (JCS)** integrates the factorized inner product prediction structure into a unified training framework, jointly optimizing the text encoder and model embeddings.

We evaluate our framework on 75 LLMs covering proprietary and open-source models with parameter scales from 0.35B to 253B. We assess interaction prediction quality across correctness, latency, and cost, downstream routing performance under varying cost constraints, few-shot adaptation to unseen models, and the interpretability of learned latent structure.

Our main contributions are summarized as follows:

- We empirically demonstrate that query-model interaction matrices in tool-calling scenarios exhibit a low-rank structure, motivating our matrix-factorization-based approach.

- We propose a dual-pathway framework that decomposes query-model interaction matrices into latent vectors for multi-objective prediction, comprising **Capability-First (CF)** and **Joint Capability-Semantic (JCS)**, enabling unified prediction of correctness, latency, and cost.

- Experiments on 75 LLMs demonstrate that our framework achieves strong multi-objective prediction performance, efficient unseen-model adaptation, parameter-efficient training, and interpretable latent representations.

## 2 Related Work

### 2.1 LLM Routing

LLM routing aims to select the optimal model for each query based on predicted performance. Early approaches frame routing as a binary decision between a small and a large model (Ding, 2024; Ong et al., 2024; Chen et al., 2023). Subsequent work has explored diverse strategies: contrastive learning for query-model matching (Chen et al., 2024), radial Transformer structures (Jin et al., 2025), heterogeneous graph-based routing (Feng et al., 2025), feature-driven routing (Agrawal and Gupta, 2025), and causal policy learning from observational data (Tsiourvas et al., 2025). Benchmarking efforts include RouterBench (Hu et al., 2024) and RouterEval (Huang et al., 2025), and PILOT (Panda et al., 2025) formulates routing as a contextual bandit problem. These methods treat routing as a classification or matching problem, relying on surface-level semantic features. Our work treats routing as a downstream application of interaction prediction, motivated by the observed low-rank structure of behavioral outcomes.

### 2.2 Tool-Calling and Agent Capability Evaluation

Tool-use capability has emerged as a critical dimension of LLM evaluation. BFCL (Patil et al., 2025) provides a comprehensive benchmark for function-calling across diverse tool schemas, while Hammer (Lin et al., 2025) focuses on fine-grained tool-use correctness. ToolACE (Liu et al., 2025) and APIBench (Patil et al., 2023) further extend evaluation to multi-tool orchestration and API interaction scenarios. These benchmarks reveal that tool-calling induces structured behavioral differences across models, providing a suitable testbed for studying capability interaction structure. We use these evaluation outcomes as input to interaction modeling rather than as endpoints for model comparison.

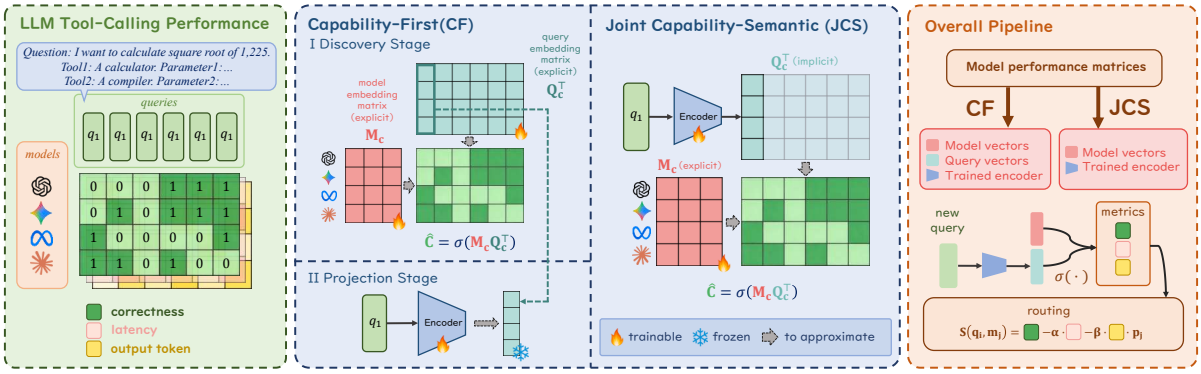



Figure 1: Overview of the latent dual-pathway framework. CF first discovers latent structure via matrix factorization (Discovery Stage) and then projects query text into this space (Projection Stage). JCS jointly optimizes the text encoder and model embeddings.

### 2.3 Collaborative Filtering and Interaction Modeling

Matrix factorization has been foundational in recommendation systems (Sarwar et al., 2000; Koren et al., 2009), where user-item rating matrices are decomposed into latent factors that capture preference structure. Extensions such as SVD++ (Koren, 2008) incorporate implicit feedback, and BPR (Rendle et al., 2009) optimizes for ranking rather than rating prediction. Neural collaborative filtering methods further enhance this paradigm with deep architectures (He et al., 2017; Wu et al., 2023). The cold-start problem, addressing new users or items with limited data, has been extensively studied in the recommendation literature (Koren, 2008; Wu et al., 2023) and is analogous to our few-shot model adaptation. Our work applies collaborative filtering techniques to LLM capability modeling: we treat queries as "users", models as "items", and behavioral outcomes as "ratings", showing that the collaborative filtering paradigm extends to structured interaction prediction in the LLM domain.

## 3 Problem Formulation

Let $\mathcal{Q} = \{q_1, \ldots, q_N\}$ denote a set of queries and $\mathcal{M} = \{m_1, \ldots, m_M\}$ denote a set of models. Each query $q_i$ is associated with a tool description specifying available tools and their schemas. For each query-model pair $(q_i, m_j)$, we observe three behavioral outcomes: correctness $c_{ij} \in \{0, 1\}$, latency $\lambda_{ij} \in \mathbb{R}^+$, and token consumption $\tau_{ij} \in \mathbb{R}^+$. We organize these into three interaction matrices of size $M \times N$, where rows correspond to models and columns to queries:

- Correctness matrix $\mathbf{C} \in \{0, 1\}^{M \times N}$ with entry $\mathbf{C}_{j,i} = c_{ij}$,
- Latency quantile matrix $\mathbf{\Lambda} \in [0, 1]^{M \times N}$ with entry $\mathbf{\Lambda}_{j,i} = Q_\lambda(\lambda_{ij})$,
- Token quantile matrix $\mathbf{T} \in [0, 1]^{M \times N}$ with entry $\mathbf{T}_{j,i} = Q_\tau(\tau_{ij})$,

Here $Q_\lambda(\cdot)$ and $Q_\tau(\cdot)$ denote empirical quantile functions that normalize raw values to $[0, 1]$:

$$Q_\lambda(\lambda) = \frac{|\{k : \lambda_{(k)} \leq \lambda\}|}{K}, \tag{1}$$

where $K$ is the total number of observed latency values and $\lambda_{(k)}$ denotes the $k$-th order statistic. This normalization addresses the heavy-tailed distributions of raw latency and token counts, which would otherwise complicate precise prediction.

The primary task of this work is query-model interaction prediction: given a query $q_i$ and model $m_j$, predict the behavioral outcomes $\hat{c}_{ij}$, $\hat{\lambda}_{ij}$, $\hat{\tau}_{ij}$. We view this as a multi-target prediction problem over query-model pairs, which can benefit from modeling approaches beyond standard text classification.

Routing is a downstream application of interaction prediction. For query $q_i$ and model $m_j$, we define the routing score as:

$$S(q_i, m_j) = \hat{c}_{ij} - \alpha \cdot \hat{\lambda}_{ij} - \beta \cdot \hat{\tau}_{ij} \cdot p_j, \tag{2}$$

where $p_j$ denotes the unit token price of model $m_j$, and $\alpha, \beta$ are trade-off hyperparameters. The routing decision selects $f(q_i) = \arg\max_{m_j \in \mathcal{M}} S(q_i, m_j)$.

## 4 Method

### 4.1 Spectral Motivation

We aim to understand whether query-model interactions admit low-dimensional structure, which would motivate matrix factorization as a modeling paradigm. We construct three interaction matrices from 75 LLMs and 2,000 tool-calling queries (correctness, latency, and token consumption; see Section B.1 for details) and perform Singular Value Decomposition (SVD) on the mean-centered matrices, and observe three key properties (see Figure 4 in Appendix C.1 for full spectral analysis).

**Rapid spectral decay.** Singular values drop sharply: the top component explains over 30% of variance, and the top 10 components capture the majority across all three matrices.

**Low effective rank.** The effective rank is 5.9 for correctness, 11.6 for latency, and 6.4 for token consumption, far below the matrix dimension of 75.

**Accurate low-rank reconstruction.** Rank-16 reconstruction achieves MSE of 0.055, 0.007, and 0.007 for the three matrices respectively. These properties provide empirical support for using matrix factorization as an inductive bias: if the interaction matrices are approximately low-rank, then factorizing them into low-dimensional latent vectors is a principled modeling choice.

### 4.2 Latent Dual-Pathway Framework

We propose a latent dual-pathway framework for capability modeling, as illustrated in Figure 1. Both pathways share the same underlying principle of representing queries and models as vectors in a latent space where their inner product predicts behavioral outcomes, but the pathways differ in how the query vectors are derived.

#### 4.2.1 Capability-First (CF)

CF adopts a two-stage design that separates capability discovery from semantic projection.

**Discovery Stage.** We perform neural matrix factorization using only interaction data, without any text encoder. Given embedding dimension $d$, we model the three interaction matrices as:

$$\begin{aligned}\hat{\mathbf{C}} &= \sigma(\mathbf{M}_c\mathbf{Q}_c^\top),\\ \hat{\mathbf{\Lambda}} &= \sigma(\mathbf{M}_\lambda\mathbf{Q}_\lambda^\top),\\ \hat{\mathbf{T}} &= \sigma(\mathbf{M}_\tau\mathbf{Q}_\tau^\top),\end{aligned} \tag{3}$$

where $\mathbf{M}_c, \mathbf{M}_\lambda, \mathbf{M}_\tau \in \mathbb{R}^{M\times d}$ are model embedding matrices, $\mathbf{Q}_c, \mathbf{Q}_\lambda, \mathbf{Q}_\tau \in \mathbb{R}^{N\times d}$ are query embedding matrices, and $\sigma(\cdot)$ is the sigmoid function. The key design principle is that the Discovery Stage extracts latent structure without relying on textual features: the query embeddings $\mathbf{Q}$ are learned purely from interaction patterns, allowing them to capture behavioral regularities that may not correspond to surface semantics.

We train the model by minimizing the following losses. For correctness, we employ binary cross-entropy (BCE) loss:

$$\mathcal{L}_{\text{corr}} = -\frac{1}{|\mathcal{D}|}\sum_{(i,j)}\left[c_{ij}\log p_{ij} + (1-c_{ij})\log(1-p_{ij})\right], \tag{4}$$

where $p_{ij}=\sigma(\mathbf{m}_c^{(j)\top}\mathbf{q}_c^{(i)})$ is the predicted correctness probability from the inner product of model and query latent vectors. For latency and token quantile matrices, we use mean squared error (MSE):

$$\mathcal{L}_{\text{lat}} = \frac{1}{|\mathcal{D}|}\sum_{(i,j)\in\mathcal{D}}\left(\sigma(\mathbf{m}_\lambda^{(j)\top}\mathbf{q}_\lambda^{(i)}) - Q_\lambda(\lambda_{ij})\right)^2, \tag{5}$$

$$\mathcal{L}_{\text{tok}} = \frac{1}{|\mathcal{D}|}\sum_{(i,j)\in\mathcal{D}}\left(\sigma(\mathbf{m}_\tau^{(j)\top}\mathbf{q}_\tau^{(i)}) - Q_\tau(\tau_{ij})\right)^2. \tag{6}$$

The global loss on the Discovery Stage is:

$$\mathcal{L}_{\text{disc}} = \mathcal{L}_{\text{corr}} + \mathcal{L}_{\text{lat}} + \mathcal{L}_{\text{tok}}. \tag{7}$$

After training, we concatenate the embeddings into joint representations: $\mathbf{M} = [\mathbf{M}_c|\mathbf{M}_\lambda|\mathbf{M}_\tau] \in \mathbb{R}^{M\times 3d}$ and $\mathbf{Q} = [\mathbf{Q}_c|\mathbf{Q}_\lambda|\mathbf{Q}_\tau] \in \mathbb{R}^{N\times 3d}$.

**Projection Stage.** We train an encoder (e.g., mDeBERTa-v3-base) to predict the query vectors $\hat{\mathbf{q}}^{(i)} \in \mathbb{R}^{3d}$ obtained from the Discovery Stage, given only the query text. A projection head maps the final hidden state to the capability space. We use LoRA (Hu et al., 2022) for parameter-efficient fine-tuning with MSE loss:

$$\mathcal{L}_{\text{proj}} = \frac{1}{N}\sum_i \|\hat{\mathbf{q}}^{(i)} - \mathbf{q}^{(i)}\|^2. \tag{8}$$

This two-stage design ensures that the latent structure is shaped by behavioral evidence: CF's procedure acts as an implicit structural regularizer through projection error, benefiting prediction across all behavioral dimensions.

**Inference.** For a new query $q$, the encoder predicts $\hat{\mathbf{q}}$, which is then multiplied with model

vectors: $\hat{\mathbf{c}} = \sigma(\mathbf{M}_c \hat{\mathbf{q}}_c)$, $\hat{\boldsymbol{\lambda}} = \sigma(\mathbf{M}_\lambda \hat{\mathbf{q}}_\lambda)$, $\hat{\boldsymbol{\tau}} = \sigma(\mathbf{M}_\tau \hat{\mathbf{q}}_\tau)$. Computational complexity is $O(M \times d)$, enabling fast inference (under 1ms per query on a single GPU).

#### 4.2.2 Joint Capability-Semantic (JCS)

JCS integrates the factorized inner product prediction structure into a unified training framework. The encoder directly predicts query vectors while maintaining learnable model embeddings:

$$\hat{\mathbf{q}}^{(i)} = \text{Projection}(\mathbf{h}_{\text{last}}^{(i)}) \in \mathbb{R}^{3d}, \tag{9}$$

$$\begin{aligned} \hat{c}_{ij} &= \hat{\mathbf{q}}_c^{(i)\top} \mathbf{m}_c^{(j)}, \\ \hat{\lambda}_{ij} &= \sigma(\hat{\mathbf{q}}_\lambda^{(i)\top} \mathbf{m}_\lambda^{(j)}), \\ \hat{\tau}_{ij} &= \sigma(\hat{\mathbf{q}}_\tau^{(i)\top} \mathbf{m}_\tau^{(j)}). \end{aligned} \tag{10}$$

For correctness, we employ BCE loss as in CF; for latency and token, we use MSE loss. The global JCS loss is:

$$\mathcal{L}_{\text{JCS}} = \mathcal{L}_{\text{corr}} + \mathcal{L}_{\text{lat}} + \mathcal{L}_{\text{tok}}. \tag{11}$$

The key distinction from CF is that query vector prediction and model embeddings are jointly optimized, allowing the text encoder to adapt to the structure of the capability space. This flexibility enables lower training loss but at the cost of requiring the text encoder to process all $N \times M$ query-model pairs during training, compared with only $N$ query-vector pairs in CF's Projection Stage, and potentially reducing regularization.

## 5 Experiments

### 5.1 Experimental Setup

**Dataset.** We construct our dataset from the BFCL v4 benchmark (Patil et al., 2025), collecting response records of 75 LLMs on function-calling queries, including correctness labels, latency, and output token counts. The benchmark covers a large and diverse set of tools spanning categories such as finance, weather, travel, and database operations. This yields 75 × 2,373 entries. Raw latency and token counts are converted to empirical quantiles within the training set. We reserve 373 queries as a held-out test set and use 2,000 queries for training.

**Model Set.** The 75 LLMs cover proprietary models (GPT-4.1, GPT-5, Claude, Gemini, Grok, Mistral, Command, DeepSeek, etc.) and open-source models (Qwen, Llama, Gemma, Hammer, xLAM, Granite, etc.); parameter scales from 0.35B to 253B; and both function-calling (FC, 36 models) and non-FC variants (39 models).

**Baselines.** For interaction prediction, we compare against: *Naive*, which predicts correctness randomly and uses 0.5 for latency and token quantiles; *KNN*, which finds $k$=5 nearest neighbors by cosine similarity and aggregates labels; *Logistic Regression*, *Random Forest*, and *XGBoost*, which use sentence transformer embeddings concatenated with one-hot model ID vectors; and *End2End*, which directly predicts correctness, latency, and token values from query text and a one-hot model ID using an encoder with three prediction heads. For downstream routing, we additionally compare against *Cosine Classifier* (Ong et al., 2024), which routes based on cosine similarity between query and model embeddings, and *RouterDC* (Chen et al., 2024), which employs dual contrastive learning. See Appendix B.2 for full baseline details and Appendix B for hyperparameters.

**Encoders.** All methods (both baselines and ours) are trained and evaluated using two text encoders: sentence transformer (all-MiniLM-L6-v2, denoted ST) (Reimers and Gurevych, 2019) and mDeBERTa-v3-base (He et al., 2021).

**Evaluation Metrics.** We evaluate interaction prediction using accuracy and F1 for correctness, and Pearson correlation and mean absolute error (MAE) for latency and token quantiles. For downstream routing, we use the ground-truth routing score:

$$\begin{aligned} S_{\text{gt}}(q_i, m_j) = \max\big(0,\ & c_{ij} - \alpha \cdot Q_\lambda(\lambda_{ij}) \\ & - \beta \cdot Q_\tau(\tau_{ij}) \cdot p_j\big), \end{aligned} \tag{12}$$

where $c_{ij} \in \{0, 1\}$ is the observed correctness (not predicted confidence).

### 5.2 Main Results

#### 5.2.1 Interaction Prediction

Table 1 presents the interaction prediction results on the held-out test set (373 queries × 75 models = 27,975 samples).

CF achieves the best overall interaction prediction across correctness, latency, and cost. mDeBERTa-CF attains the highest accuracy (0.7727) and ST-CF closely follows (0.7699), both outperforming JCS, End2End, and ML baselines. For latency and token prediction, CF also achieves the best or near-best performance across all metrics. The matrix factorization structure provides

Table 1: Interaction prediction results on the held-out test set. We report accuracy, F1, token MAE, token correlation, latency MAE, and latency correlation. Best results within each encoder group are **boldfaced**.

| Method | Encoder | Acc. | F1 | Tok. MAE↓ | Tok. Corr | Lat. MAE↓ | Lat. Corr |
|---|---|---|---|---|---|---|---|
| Naive | - | 0.5000 | 0.5895 | 0.2480 | – | 0.2501 | – |
| KNN | ST | 0.7100 | 0.7800 | 0.1800 | 0.6500 | 0.1700 | 0.6200 |
| Logistic Reg. | ST | 0.7330 | 0.8332 | 0.1336 | 0.7983 | 0.1087 | 0.8575 |
| Random Forest | ST | 0.7275 | 0.8389 | 0.1596 | 0.7079 | 0.1659 | 0.6886 |
| XGBoost | ST | 0.7422 | 0.8439 | 0.1440 | 0.8045 | 0.1305 | 0.8569 |
| End2End | ST | 0.7567 | 0.8403 | 0.1322 | 0.8062 | 0.1091 | 0.8619 |
| End2End | mDeBERTa | 0.7430 | 0.8301 | 0.1316 | 0.8037 | 0.1089 | 0.8586 |
| JCS (Ours) | ST | 0.7579 | 0.8389 | 0.1250 | 0.8116 | 0.1023 | 0.8712 |
| JCS (Ours) | mDeBERTa | 0.7574 | 0.8372 | 0.1251 | 0.8136 | 0.1019 | 0.8717 |
| CF (Ours) | ST | 0.7699 | **0.8531** | 0.1142 | 0.8341 | 0.0940 | 0.8840 |
| CF (Ours) | mDeBERTa | **0.7727** | 0.8514 | **0.1130** | **0.8382** | **0.0933** | **0.8847** |

a stronger inductive bias than direct prediction, which aligns with the spectral analysis.

ML baselines perform competitively on correctness but substantially worse on continuous prediction. XGBoost achieves 0.7422 accuracy but shows substantially worse token/latency correlation (0.8045 and 0.8569, compared with 0.8382 and 0.8847 for mDeBERTa-CF), limiting their effectiveness for cost-aware routing.

#### 5.2.2 Downstream Routing Results

We evaluate routing as a downstream application of interaction prediction. Table 2 reports routing scores and accuracies under a sub-sampling protocol (75 models randomly partitioned into groups of 10, 100 trials); see Appendix C.2 for additional experimental details.

Our interaction prediction model jointly predicts correctness, latency, and token consumption, enabling cost-aware routing. CF achieves the best routing scores (mDeBERTa-CF: 0.859 at $\alpha{=}\beta{=}0$), while JCS and End2End achieve the best routing accuracy under cost-aware settings (mDeBERTa-JCS: 0.809 at $\alpha{=}\beta{=}0.2$). Both substantially outperform existing routing baselines. Results for the general case $\alpha \neq \beta$ are provided in Appendix C.2.

In contrast to our framework, the routing baselines above do not leverage cost information: Cosine Classifier and RouterDC show nearly constant routing accuracy across cost weights, with scores dropping sharply as $\alpha{=}\beta$ increases, whereas our methods maintain strong performance by jointly predicting correctness, latency, and cost.

### 5.3 Few-Shot New Model Adaptation

A distinctive advantage of the collaborative filtering paradigm is new-model adaptation: new models can be incorporated by learning only their embedding vector while keeping the query space fixed, analogous to how new items are added in recommendation systems. The observed low-rank structure of the interaction matrices implies that model behavior can be captured by a low-dimensional embedding, enabling effective adaptation from very few samples.

Performance improves monotonically with more data but with diminishing returns: the most significant gain occurs between 100 and 250 samples (+2.04% accuracy), and only +0.21% from 500 to 1,000. The fixed query vectors provide a strong inductive bias that constrains the model embedding to a region consistent with the pre-learned capability space, preventing overfitting even with limited data.

### 5.4 Ablation Studies

We conduct ablation studies on mDeBERTa-CF to examine the impact of key design choices. Table 4 summarizes the results.

**Embedding dimension.** We test the hidden dimension of embeddings $d \in \{4, 8, 16, 32\}$. Performance slightly increases for $d \leq 16$ (accuracy varies only 0.46%, 0.7681–0.7727), but degrades at $d{=}32$, suggesting overfitting when the embedding dimension exceeds the intrinsic rank. This is consistent with our spectral analysis showing effective rank of 5.91–11.56.

**Decomposition method.** Neural MF substantially outperforms SVD on continuous prediction (token correlation: 0.8382 compared with 0.0737). This gap is expected because SVD's unsupervised decomposition optimizes for reconstruction rather than prediction, and its unbounded outputs are poorly suited for quantile-valued targets.

**Input representation.** Tool descriptions are

Table 2: Downstream routing results under sub-sampling (10 models/group, 100 trials). Routing decision is confidence-based; routing score uses real latency/token. Best results within each encoder group are **boldfaced**.

| Method | Encoder | Routing Score ($\alpha = \beta$) | | | | | | Routing Accuracy ($\alpha = \beta$) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 0.2 | 0.4 | 0.6 | 0.8 | 1.0 | 0 | 0.2 | 0.4 | 0.6 | 0.8 | 1.0 |
| Oracle | - | 0.974 | 0.908 | 0.848 | 0.789 | 0.732 | 0.681 | 0.974 | 0.971 | 0.968 | 0.963 | 0.955 | 0.948 |
| Cosine Class. | ST | 0.825 | 0.601 | 0.435 | 0.295 | 0.197 | 0.143 | 0.825 | **0.824** | **0.824** | **0.825** | **0.824** | **0.823** |
| RouterDC | ST | 0.763 | 0.557 | 0.424 | 0.310 | 0.220 | 0.176 | 0.763 | 0.763 | 0.762 | 0.764 | 0.766 | 0.763 |
| KNN | ST | 0.795 | 0.684 | 0.637 | 0.592 | 0.545 | 0.502 | 0.795 | 0.731 | 0.733 | 0.733 | 0.731 | 0.731 |
| End2End | ST | 0.844 | **0.737** | 0.665 | 0.610 | 0.558 | 0.517 | 0.844 | 0.813 | 0.786 | 0.769 | 0.754 | 0.746 |
| End2End | mDeBERTa | 0.844 | 0.730 | 0.662 | 0.605 | 0.556 | 0.511 | 0.844 | 0.799 | 0.776 | 0.760 | 0.746 | 0.737 |
| JCS (Ours) | ST | 0.837 | 0.733 | 0.664 | 0.609 | 0.557 | 0.516 | 0.837 | 0.812 | 0.787 | 0.765 | 0.745 | 0.734 |
| JCS (Ours) | mDeBERTa | 0.839 | 0.735 | **0.667** | 0.613 | 0.563 | 0.519 | 0.839 | 0.809 | 0.791 | 0.778 | 0.761 | 0.752 |
| CF (Ours) | ST | 0.852 | 0.721 | 0.661 | 0.611 | 0.564 | 0.524 | 0.852 | 0.780 | 0.765 | 0.756 | 0.749 | 0.744 |
| CF (Ours) | mDeBERTa | **0.859** | 0.723 | 0.665 | **0.617** | **0.569** | **0.524** | **0.859** | 0.782 | 0.771 | 0.767 | 0.761 | 0.754 |

Table 3: Few-shot adaptation of mDeBERTa-CF (10-run average). Frozen query space with learnable model embedding enables collaborative filtering-style cold-start.

| Train Size | Acc. | F1 | L-MAE↓ | T-MAE↓ |
|---|---|---|---|---|
| 100 | 0.7779 | 0.8185 | 0.0824 | 0.0939 |
| 250 | 0.7983 | 0.8356 | 0.0763 | 0.0868 |
| 500 | 0.8037 | 0.8401 | 0.0750 | 0.0852 |
| 1000 | **0.8058** | **0.8413** | **0.0745** | **0.0845** |

Table 4: Ablation study on mDeBERTa-CF. We examine embedding dimension, decomposition method, and input representation (Q=Question, TD=Tool Description, P=Parameter schema).

| Aspect | Setting | Acc. | F1 | T-MAE↓ | T-Corr | L-MAE↓ | L-Corr |
|---|---|---|---|---|---|---|---|
| Dim | 4 | 0.7681 | 0.8466 | 0.1145 | 0.8327 | 0.0939 | 0.8843 |
| | 8 | 0.7719 | 0.8513 | 0.1139 | 0.8365 | 0.0934 | **0.8851** |
| | 16 | **0.7727** | **0.8514** | **0.1130** | **0.8382** | **0.0933** | 0.8847 |
| | 32 | 0.7645 | 0.8486 | 0.1157 | 0.8349 | 0.0957 | 0.8836 |
| Decomp. | Neural MF | **0.7727** | **0.8514** | **0.1130** | **0.8382** | **0.0933** | **0.8847** |
| | SVD | 0.7265 | 0.8296 | 0.2726 | 0.0737 | 0.3947 | 0.0294 |
| Input | Q | 0.7514 | 0.8346 | 0.1167 | 0.8264 | 0.0985 | 0.8733 |
| | Q+TD+P | 0.7560 | 0.8357 | 0.1168 | 0.8277 | 0.0956 | 0.8807 |
| | Q+TD | **0.7727** | **0.8514** | **0.1130** | **0.8382** | **0.0933** | **0.8847** |

essential (+2.13% accuracy over question-only), confirming that interaction depends on capability-query matching, not just question semantics. Adding full parameter specifications does not improve over tool descriptions alone, possibly because detailed parameter schemas introduce redundant or distracting information that obscures the relevant signal.

## 5.5 Capability Interpretability

### 5.5.1 FC versus Non-FC Model Clustering

K-means clustering on model vectors achieves ARI (Adjusted Rand Index) = 0.136 with respect to FC (function-calling) versus non-FC labels, higher than model size (ARI = 0.029) and open/proprietary type (ARI = 0.007). Although the absolute ARI values are low, the consistent relative ordering across partitions suggests that the learned representations are more aligned with functional capability than with model scale or provenance (Figure 2a).

### 5.5.2 Question Type Structure

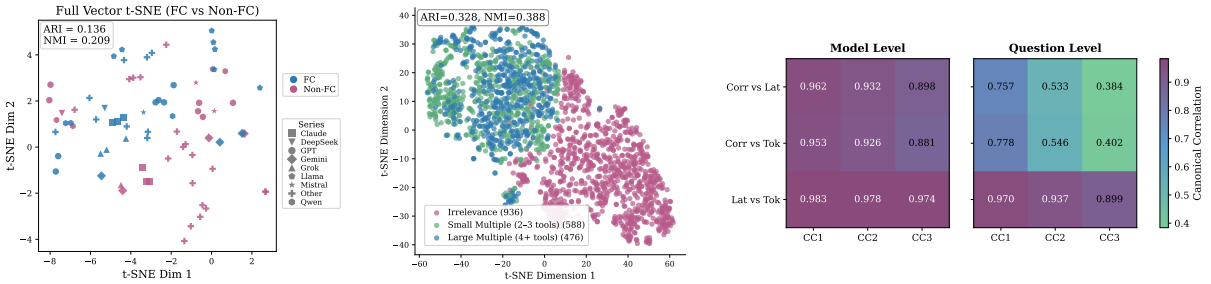
The learned question vectors also exhibit interpretable structure. K-means clustering ($k$=3) on the full question vectors yields ARI = 0.328 and NMI (Normalized Mutual Information) = 0.388 with respect to the two ground-truth question categories: Multiple (1,253 questions requiring selection from multiple candidate tools) and Irrelevance (1,120 questions where no tool should be invoked). The clustering partially separates queries that require active tool selection from those requiring abstention, suggesting that the latent space captures some of the distinction between tool-matching and irrelevance-detection behaviors (Figure 2b).

### 5.5.3 Cross-Task Correlation via CCA

Latency and token vectors are highly correlated at the model level ($CC_1$, first canonical correlation = 0.983), while correctness is partially independent of cost (correctness-latency $CC_1$ drops from 0.962 at model level to 0.757 at question level). This is consistent with our multi-task design: the three behavioral dimensions capture related but distinct aspects of query-model interaction (Figure 2c).

## 5.6 Data Efficiency

CF is the most data-efficient: the Discovery Stage requires only the $M \times N$ interaction matrix (no text encoding), and the Projection Stage trains on just $N = 2{,}000$ query-vector pairs. At inference, CF and JCS require only 373 forward




(a) (b) (c)

Figure 2: Latent space interpretability. (a) t-SNE of model vectors: FC and non-FC models yield the highest ARI (0.136) among categorical partitions. (b) t-SNE of question vectors colored by type: K=3 clustering achieves ARI = 0.328, NMI = 0.388. (c) Canonical correlation analysis between task sub-vectors at model and question levels.

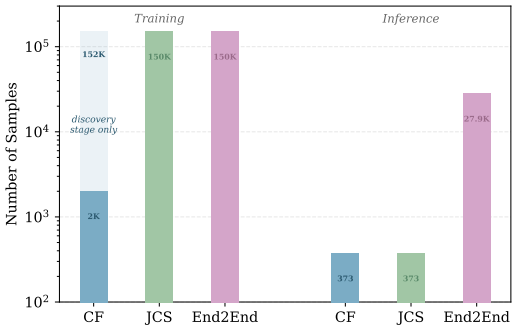



Figure 3: Data efficiency comparison. CF's Projection Stage needs only 2,000 query-vector pairs; at inference, CF and JCS require only 373 forward passes compared with 27,975 for End2End.

passes compared with 27,975 for End2End.

## 6 Discussion

**Latent structure of model capabilities.** Our spectral analysis reveals that query-model interactions reside in a low-dimensional space. The learned model embeddings show stronger alignment with functional capability (FC versus non-FC) than with surface attributes (size, vendor), though the low absolute clustering scores indicate that simple external labels have limited correlation with latent features.

**Interaction-driven versus semantics-driven prediction.** The dominant paradigm in LLM routing is semantics-driven: encode the query, then predict outcomes. Our results suggest the value of interaction-driven prediction: first discover the behavioral structure from interaction data, then project language into this structure. Interaction data directly reflects capability, while language provides an indirect signal that may not fully capture behavioral patterns.

**Collaborative filtering for LLM capability modeling.** The analogy to recommendation systems is direct: queries correspond to users, models to items, and behavioral outcomes to ratings. The cold-start problem maps to new model adaptation, and the low-rank assumption maps to the latent structure. This connection suggests the possibility of transferring advances in collaborative filtering to the LLM capability domain.

**Scalability.** Our current framework handles 75 models; scaling to hundreds or thousands of models raises computational challenges. However, the low-rank structure suggests that approximate methods can maintain efficiency.

## 7 Conclusion

We present a latent dual-pathway framework for LLM tool-calling prediction. Our key finding is that query-model interactions empirically exhibit a low-rank structure, suggesting that capability prediction can benefit from structured interaction modeling. The proposed dual-pathway framework, comprising **Capability-First (CF)** and **Joint Capability-Semantic (JCS)**, separates behavioral structure discovery from semantic mapping. Experiments on 75 LLMs demonstrate that CF achieves the best overall interaction prediction across correctness, latency, and cost, while JCS achieves the best routing accuracy under cost-aware settings. Further experiments demonstrate that our method enables efficient few-shot model generalization, parameter-efficient training, and interpretable latent structure.

## Limitations

While our proposed framework demonstrates compelling efficacy, several intertwined limitations

remain, which simultaneously chart promising avenues for future research. First, our empirical evaluation is primarily anchored on the BFCL v4 benchmark, focusing on function-calling scenarios; scaling this paradigm to more heterogeneous task modalities, such as complex code synthesis and multi-step mathematical reasoning, stands as an immediate priority for our next-stage development. Second, the current routing objective relies on a simplified linear combination of correctness, latency, and cost. To better align with sophisticated production demands, future endeavors will investigate non-linear optimization strategies and learnable preference models to capture more nuanced user trade-offs. Lastly, the current architecture is inherently constrained to single-turn interactions. Extending the routing infrastructure to orchestrate stateful, multi-turn tool usage represents a non-trivial yet essential horizon, where maintaining context consistency across disparate models will be systematically explored.

## References


Shubham Agrawal and Prasang Gupta. 2025. LLM-Rank: Understanding LLM strengths for model routing. *arXiv preprint arXiv:2510.01234*.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32.

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

Shuhao Chen, Weisen Jiang, Baijiong Lin, James T. Kwok, and Yu Zhang. 2024. RouterDC: Query-based router by dual contrastive learning for assembling large language models. *arXiv preprint arXiv:2409.19886*.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794.

David R. Cox. 1958. The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2):215–242.

Jiayu Ding. 2024. Hybrid LLM: Cost-efficient and quality-aware query routing. *arXiv preprint arXiv:2404.14618*.

Tao Feng, Yanzhen Shen, and Jiaxuan You. 2025. GraphRouter: A graph-based router for LLM selections. In *International Conference on Learning Representations*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTa: Decoding-enhanced BERT with disentangled attention. In *International Conference on Learning Representations*.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. *Proceedings of the 26th International Conference on World Wide Web*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Yixin Hu, Wangyue Li, and Litong Gao. 2024. Routerbench: A benchmark for multi-llm routing system. *arXiv preprint arXiv:2409.05694*.

Zhongzhan Huang, Guoming Ling, Yupei Lin, Yandong Chen, and Shanshan Zhong. 2025. Routereval: A comprehensive benchmark for routing LLMs to explore model-level scaling up in LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2025*.

Ruihan Jin, Pengpeng Shao, Zhengqi Wen, Jinyang Wu, Mingkuan Feng, Shuai Zhang, and Jianhua Tao. 2025. Radialrouter: Structured representation for efficient and robust large language models routing. In *Findings of the Association for Computational Linguistics: EMNLP 2025*.

Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37.

Qiqiang Lin, Muning Wen, Qiuying Peng, Guanyu Nie, Junwei Liao, Jun Wang, Xiaoyun Mo, Jiamu Zhou, Cheng Cheng, Yin Zhao, Jun Wang, and Weinan Zhang. 2025. Hammer: Robust function-calling for on-device language models via function masking. In *International Conference on Learning Representations*.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong Wang, Yuxian Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Xinzhi Wang, Yong Liu, Yasheng Wang, and 8 others. 2025. ToolACE: Winning the points of LLM function calling. In *International Conference on Learning Representations*.

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

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. 2024. RouteLLM: Learning to route LLMs with preference data. *arXiv preprint arXiv:2406.18665*.

Pranoy Panda, Raghav Magazine, Chaitanya Devaguptapu, Sho Takemori, and Vishal Sharma. 2025. Adaptive LLM routing under budget constraints.

Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. Gorilla: Large language model connected with massive APIs. *arXiv preprint arXiv:2305.15334*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*.

Badrul Sarwar, Yehuda Koren, Robert Bell, and Chris Volinsky. 2000. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295.

Asterios Tsiourvas, Wei Sun, and Georgia Perakis. 2025. Causal LLM routing: End-to-end regret minimization from observational data. In *Advances in Neural Information Processing Systems*.

Le Wu, Xiangnan He, Xiang Wang, Kun Zhang, and Meng Wang. 2023. A survey on accuracy-oriented neural recommendation: From collaborative filtering to information-rich recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 35(5):4425–4445.

# A Theoretical Analysis

## A.1 Interpretation of CF Regularization Effect

CF's two-stage procedure acts as an implicit structural regularizer. Let $(Q^*, M^*)$ be the Discovery Stage solution and $\hat{\theta}$ the Projection Stage solution. For any query $i$ and model $j$:

$$\mathcal{L}\left(R_{ij}, f_{\hat{\theta}}(x_i) \cdot M_j^*\right) \leq \mathcal{L}\left(R_{ij}, Q_i^* \cdot M_j^*\right) + L \cdot \|f_{\hat{\theta}}(x_i) - Q_i^*\| \cdot \|M_j^*\|. \quad (13)$$

This bound reveals the key relationship:

- When the text encoder perfectly fits the query embeddings ($\|f_\theta(X) - Q^*\| \to 0$), CF approaches the performance of optimal low-rank decomposition.
- When the fit is imperfect, the projection error $\epsilon = \|f_\theta(X) - Q^*\|^2$ effectively restricts the hypothesis class, reducing variance at the cost of increased bias.

By the bias-variance tradeoff, this regularization particularly benefits continuous cost predictions that dominate the routing score. This may explain why CF achieves the best routing scores despite JCS's potential to achieve lower training loss.

# B Experimental Setup

## B.1 Dataset Details

### B.1.1 Query Examples

Table 5 presents representative queries from the BFCL v4 benchmark, illustrating the actual input format provided to each model.

### B.1.2 Data Source and Processing

Our dataset is derived from the BFCL v4 benchmark (Patil et al., 2025). We extract model response data from the BFCL-Result repository, using the snapshot dated 2025-12-16, which contains inference results for over 100 LLMs. From these, we select 75 models that have complete coverage of all evaluation questions, ensuring a fully observed interaction matrix. We retain only single-turn scenarios and exclude multi-turn conversations and simple-category questions. The remaining questions fall into two categories:

**Multiple** (1,253 unique questions): Each question is accompanied by multiple candidate tools, and the model must select and invoke the correct subset.

**Irrelevance** (1,120 unique questions): Each question is paired with tools that are irrelevant to the query. A response is correct if the model refrains from invoking any tool.

### B.1.3 Model List and Pricing

Table 6 lists all 75 models and their per-million-token output prices (USD). For proprietary models, prices are sourced from official API pricing pages as of December 2025. For open-source models, we estimate the cost based on GPU-hour pricing: assuming an A100 GPU at $2.21/hour, we

Table 5: Representative query examples from BFCL v4. Each query consists of a user question and a set of candidate tools with their descriptions and parameter schemas. The model must select and invoke the correct tool(s) or refrain from invoking any tool.

| Type | Question | Available Tools | Expected Behavior |
|---|---|---|---|
| Multiple | *Update my latte to a large size with coconut milk and make it extra sweet? Make it served "boiling hot" as the special request.* | *ChaFod: Changes the food item based on the customer's request. Params: foodItem, newIngredients, removeIngredients, specialInstructions.* *ChaDri.change_drink: Modifies the existing drink order. Params: drink_id, new_preferences.* | Invoke `ChaDri.change_drink` with `drink_id="latte"` |
| Multiple | *Book a flight from Seattle to Boston with American Airlines and book a hotel in Boston for 4 nights.* | *flight_book: Book a flight for a specific route and airlines. Params: _from, to, airlines.* *hotel_book: Book a hotel for a specific location for the number of nights. Params: location, nights.* | Invoke both `flight_book` and `hotel_book` with appropriate parameters |
| Irrelevance | *Can you provide the address for latitude 37.42 and longitude -122.08 using the Geocoding API?* | *requests.get: Sends a GET request to the specified URL to retrieve data. Params: url, headers, timeout, params.* | Refrain from invoking any tool (no relevant tool available) |
| Irrelevance | *Write a Python script to change filenames in a folder with their date of creation.* | *requests.get: Sends a GET request to the specified URL to retrieve data. Params: url, headers, timeout, params.* | Refrain from invoking any tool (no relevant tool available) |

compute the per-token cost from the model's inference throughput. The routing score formula $S = \hat{c} - \alpha\hat{\lambda} - \beta\hat{\tau} \cdot p$ treats price as a plug-in parameter; users can substitute any pricing scheme without retraining the model.

## B.2 Baselines

### B.2.1 Basic Prediction Baselines

We implement three ML baselines and one neural baseline. All ML baselines take as input a feature vector concatenating Sentence Transformer embeddings (384d) with a one-hot model ID vector (75d), yielding 459-dimensional input features.

**Logistic Regression.** (Cox, 1958) A linear classification method that models the probability of binary outcomes using the logistic function. For correctness: logistic regression with max iteration 1,000, $C$=1.0, LBFGS solver. For latency/token: Ridge regression with $\alpha$=1.0, which adds L2 regularization to linear regression.

**Random Forest.** (Breiman, 2001) An ensemble learning method that constructs multiple decision trees during training and outputs the mode of the class predictions (for classification) or mean prediction (for regression). We use 100 estimators with max depth 20.

**XGBoost.** (Chen and Guestrin, 2016) A gradient boosting framework that builds decision trees sequentially, with each new tree correcting errors from previous trees. We use 100 estimators, max depth 6, and learning rate 0.1.

**End2End.** An end-to-end neural approach that directly predicts correctness, latency, and token values from query text and a one-hot model ID using an encoder with three prediction heads. We use the same mDeBERTa-v3-base encoder and sentence transformer encoder as our methods, with LoRA fine-tuning. The model learns to map query-model pairs directly to behavioral outcomes without explicit factorization.

### B.2.2 Routing Baselines

**Cosine Classifier** (Ong et al., 2024). A routing method that learns model embeddings and computes routing probabilities as normalized cosine similarity between query and model embeddings. It uses mDeBERTa-v3-base encoder with learnable model embeddings, where routing probability = cosine similarity / temperature. Trained with BCEWithLogitsLoss, AdamW (lr=5e-5, weight decay=0.01, batch size=16, 10 epochs, cosine annealing).

**RouterDC** (Chen et al., 2024). A dual contrastive learning approach for routing that optimizes two contrastive losses: Sample-LLM loss treats top-K scoring models as positive examples for each query, while Sample-Sample loss uses KMeans-clustered queries as positives. It

Table 6: Model list and corresponding prices (USD per million output tokens).

| Model | Price | Model | Price | Model | Price | Model | Price |
|---|---|---|---|---|---|---|---|
| claude-opus-4-5-FC | 15.00 | claude-opus-4-5 | 15.00 | gpt-5.2-FC | 7.88 | gpt-5.2 | 7.88 |
| claude-sonnet-4-5-FC | 9.00 | claude-sonnet-4-5 | 9.00 | grok-4-FC | 9.00 | grok-4 | 9.00 |
| command-a-03-FC | 6.25 | command-a-reasoning-FC | 6.25 | gpt-4.1-FC | 5.00 | gpt-4.1 | 5.00 |
| Salesforce_Llama-xLAM-2-70b | 4.81 | Llama-3.3-70B-FC | 4.81 | gemini-3-pro-FC | 3.13 | gemini-3-pro | 3.13 |
| claude-haiku-4-5-FC | 3.00 | claude-haiku-4-5 | 3.00 | gpt-5-mini-FC | 1.13 | gpt-5-mini | 1.13 |
| Qwen3-32B-FC | 1.20 | Qwen3-32B | 1.20 | gemma-3-27b | 1.20 | Salesforce_xLAM-2-32b | 1.20 |
| katanemo_Arch-32B | 1.20 | mistral-large-FC | 1.60 | mistral-large | 1.60 | mistral-medium | 1.20 |
| kimi-k2-FC | 1.37 | DeepSeek-V3.2-FC | 0.68 | DeepSeek-V3.2-thinking | 0.68 | glm-4.6-FC | 0.68 |
| Nanbeige3.5-Pro-FC | 0.68 | Llama-4-Maverick-FC | 0.72 | Llama-4-Scout-FC | 0.72 | granite-20b | 0.72 |
| gpt-4.1-mini-FC | 1.00 | gpt-4.1-mini | 1.00 | gemini-2.5-flash-FC | 0.38 | gemini-2.5-flash | 0.38 |
| Hammer2.1-7b | 0.36 | grok-4-1-fast-FC | 0.35 | grok-4-1-reasoning-FC | 0.35 | Qwen3-4B-FC | 0.24 |
| Qwen3-4B | 0.24 | gpt-5-nano-FC | 0.23 | gpt-5-nano | 0.23 | gpt-4.1-nano-FC | 0.25 |
| gpt-4.1-nano | 0.25 | gemini-2.5-flash-lite-FC | 0.25 | gemini-2.5-flash-lite | 0.25 | command-r7b-FC | 0.25 |
| Qwen3-8B-FC | 0.48 | Qwen3-8B | 0.48 | BitAgent-8B | 0.48 | ToolACE-2-8B | 0.48 |
| Salesforce_Llama-xLAM-2-8b | 0.48 | Llama-3.1-8B | 0.48 | granite-3.1-8b | 0.48 | granite-3.2-8b | 0.48 |
| gemma-3-12b | 0.48 | microsoft_phi-4 | 0.48 | gemma-3-4b | 0.24 | Qwen3-4B-FC | 0.24 |
| Hammer2.1-3b | 0.089 | Salesforce_xLAM-2-3b | 0.089 | katanemo_3B | 0.089 | Llama-3.2-3B-FC | 0.089 |
| Nanbeige4-3B | 0.089 | Hammer2.1-1.5b | 0.056 | katanemo_1.5B | 0.056 | Hammer2.1-0.5b | 0.014 |
| Salesforce_xLAM-2-1b | 0.044 | gemma-3-1b | 0.044 | Llama-3.2-1B-FC | 0.044 | granite-4.0-350m | 0.014 |

uses mDeBERTa-v3-base encoder and is trained with AdamW (lr=5e-5, batch size=64, 10 epochs, temperature=1.0, top-K=3).

**KNN.** A non-parametric routing method that encodes training queries with Sentence Transformer, finds $k$=5 nearest neighbors by cosine similarity for each test query, and aggregates their labels to make routing decisions.

### B.3 Training Hyperparameters

Table 7 summarizes the training hyperparameters for each method.

Table 7: Training hyperparameters.

| Parameter | CF | JCS | End2End |
|---|---|---|---|
| *Discovery Stage: Matrix Factorization* | | | |
| Embedding dim | 16 | – | – |
| Batch size | 1024 | – | – |
| Learning rate | 0.001 | – | – |
| Correctness epochs | 200 | – | – |
| Latency/Token epochs | 120 | – | – |
| *Projection Stage / Joint Training* | | | |
| Base encoder | ST (all-MiniLM-L6-v2) / mDeBERTa-v3-base | | |
| LoRA $r$ | 16 | 16 | 16 |
| LoRA $\alpha$ | 32 | 32 | 32 |
| LoRA dropout | 0.1 | 0.1 | 0.1 |
| Learning rate | 2e-4 | 2e-4 | 2e-4 |
| Batch size | 16 | 16 | 16 |
| Epochs | 10 | 1 | 1 |
| Max length | 512 | 512 | 512 |
| Threshold | 0.5 | 0.5 | 0.5 |

### B.4 Routing Evaluation Protocol

We adopt a sub-sampling protocol: for each trial, 75 models are randomly partitioned into groups of 10, and within each group the router selects the best model. This is repeated over 100 trials with different random partitions. To ensure fair comparison across different encoder settings, the same random partitions are used for all methods within each experiment.

### B.5 Generalization Evaluation Protocol

To evaluate generalization to unseen models, we introduce 5 new models that are not part of the original 75-model training set. For each of these unseen models, we collect 1,000 query responses for evaluation. The original 75 models and 2,000 training queries are used for training. To ensure fair comparison across different training sizes (100, 250, 500, 1000), we employ a nested training set design: the training sets are nested such that the 100-sample set is a subset of the 250-sample set, which is a subset of the 500-sample set, which is a subset of the 1000-sample set. The query vector predictor is frozen after the Projection Stage training; only the new model's embedding vector is learned.

## C Additional Experimental Results

### C.1 Spectral Analysis Details

Figure 4 shows the complete singular value curves for all three interaction matrices, including the first 50 components. Figure 5 presents the variance explained per component, and Figure 6 shows the cumulative variance explained.

The effective rank provides a continuous measure of the intrinsic dimensionality of a matrix:

$$\text{erank}(\mathbf{A}) = \frac{(\sum_i \sigma_i^2)^2}{\sum_i \sigma_i^4}. \tag{14}$$

For our interaction matrices: Correctness erank = 5.91, Latency erank = 11.56, Token erank = 6.40.

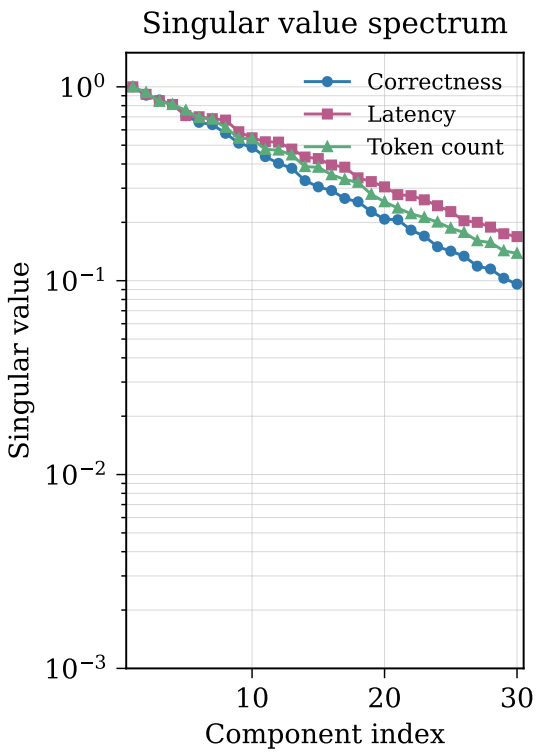



Figure 4: Singular value spectrum of the three interaction matrices (log scale). All matrices exhibit rapid spectral decay.

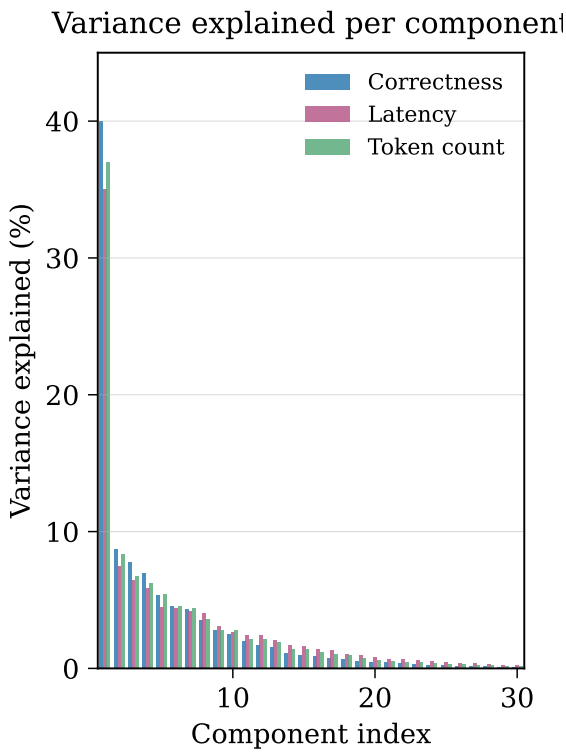



Figure 5: Variance explained per component for correctness, latency, and token matrices.

These values are far below the full rank of 75, confirming strong low-rank structure.

Table 8 shows the reconstruction MSE at various ranks.

Table 8: Reconstruction MSE at different ranks (mean-centered matrices).

| Matrix | k=1 | k=3 | k=5 | k=10 | k=16 | k=20 |
|---|---|---|---|---|---|---|
| Correctness | 0.1121 | 0.0909 | 0.0824 | 0.0679 | 0.0552 | 0.0483 |
| Latency | 0.0183 | 0.0129 | 0.0113 | 0.0087 | 0.0067 | 0.0057 |
| Token | 0.0222 | 0.0137 | 0.0116 | 0.0088 | 0.0069 | 0.0058 |

## C.2 Additional Routing Settings

Figure 7 supplements the main routing results in Table 2, presenting routing score and accuracy curves as $\alpha = \beta$ varies continuously from 0 to 1. Our methods substantially outperform routing baselines across all cost weights, confirming that the advantage observed at discrete points in Table 2 holds consistently across the entire tradeoff spectrum.

### C.2.1 Routing Results with $\alpha \neq \beta$

Table 9 and Table 10 extend the routing evaluation to the general case $\alpha \neq \beta$, where the cost weights for latency and token consumption differ. Our methods consistently outperform routing baselines across all asymmetric cost settings.

### C.2.2 Stricter Evaluation Settings

A critical concern is whether our models merely exploit the FC versus non-FC distinction as a spurious signal. We address this through more stringent evaluation protocols.

**FC-Only and Non-FC-Only Evaluation.** Table 11 shows routing performance when selecting from homogeneous capability groups. Both settings indicate that our model learns fine-grained capability distinctions beyond simple FC versus non-FC binary classification.

**Same-Family and Same-Size Evaluation.** Table 12 shows routing and prediction accuracy when the candidate pool is restricted to models within the same family or size category. Our method maintains strong performance, suggesting that it captures capability differences beyond family or scale shortcuts.

Table 9: Routing score under $\alpha \neq \beta$. Each column header shows $(\alpha, \beta)$.

| **Method** | **Enc.** | $\alpha > \beta$ (latency-focused) | | | | $\alpha < \beta$ (token-focused) | | | | Mixed | | Extreme | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | (.2,0) | (.4,0) | (.6,0) | (.8,0) | (0,.2) | (0,.4) | (0,.6) | (0,.8) | (.2,.4) | (.4,.2) | (.5,1) | (1,.5) |
| Cosine Class. | ST | 0.721 | 0.621 | 0.520 | 0.417 | 0.753 | 0.706 | 0.674 | 0.654 | 0.638 | 0.608 | 0.502 | 0.388 |
| RouterDC | ST | 0.675 | 0.591 | 0.506 | 0.419 | 0.702 | 0.680 | 0.659 | 0.634 | 0.600 | 0.548 | 0.477 | 0.337 |
| KNN | ST | 0.719 | 0.694 | 0.667 | 0.641 | 0.718 | 0.709 | 0.699 | 0.688 | 0.669 | 0.652 | 0.580 | 0.538 |
| End2End | ST | **0.774** | 0.725 | 0.686 | 0.654 | 0.800 | 0.780 | **0.763** | 0.744 | 0.717 | 0.683 | 0.601 | 0.550 |
| End2End | mDeBERTa | 0.765 | 0.720 | 0.684 | 0.649 | **0.802** | 0.777 | 0.753 | 0.734 | 0.712 | 0.677 | 0.595 | 0.546 |
| JCS (Ours) | ST | 0.768 | 0.724 | 0.684 | 0.652 | 0.795 | 0.775 | 0.759 | 0.741 | 0.714 | 0.682 | 0.602 | 0.548 |
| JCS (Ours) | mDeBERTa | 0.773 | **0.728** | **0.692** | 0.659 | 0.800 | **0.780** | 0.759 | **0.744** | **0.717** | **0.685** | 0.604 | 0.553 |
| CF (Ours) | ST | 0.755 | 0.719 | 0.685 | 0.654 | 0.773 | 0.758 | 0.744 | 0.730 | 0.701 | 0.679 | 0.603 | **0.556** |
| CF (Ours) | mDeBERTa | 0.758 | 0.723 | 0.691 | **0.660** | 0.782 | 0.764 | 0.750 | 0.736 | 0.707 | 0.682 | **0.608** | 0.555 |

Table 10: Routing accuracy under $\alpha \neq \beta$. Each column header shows $(\alpha, \beta)$.

| **Method** | **Enc.** | $\alpha > \beta$ (latency-focused) | | | | $\alpha < \beta$ (token-focused) | | | | Mixed | | Extreme | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | (.2,0) | (.4,0) | (.6,0) | (.8,0) | (0,.2) | (0,.4) | (0,.6) | (0,.8) | (.2,.4) | (.4,.2) | (.5,1) | (1,.5) |
| Cosine Class. | ST | 0.823 | 0.825 | 0.824 | 0.825 | 0.767 | 0.724 | 0.697 | 0.681 | 0.720 | 0.767 | 0.673 | 0.705 |
| RouterDC | ST | 0.760 | 0.764 | 0.762 | 0.760 | 0.725 | 0.713 | 0.690 | 0.668 | 0.703 | 0.727 | 0.645 | 0.698 |
| KNN | ST | 0.746 | 0.747 | 0.746 | 0.747 | 0.728 | 0.728 | 0.726 | 0.723 | 0.729 | 0.733 | 0.727 | 0.731 |
| End2End | ST | **0.827** | 0.805 | 0.789 | 0.779 | **0.823** | **0.813** | **0.807** | **0.800** | **0.805** | 0.790 | 0.768 | 0.751 |
| End2End | mDeBERTa | 0.812 | 0.794 | 0.780 | 0.767 | 0.821 | 0.808 | 0.794 | 0.784 | 0.794 | 0.778 | 0.754 | 0.739 |
| JCS (Ours) | ST | 0.825 | 0.809 | 0.794 | 0.781 | 0.820 | 0.809 | 0.804 | 0.799 | 0.804 | 0.792 | 0.763 | 0.743 |
| JCS (Ours) | mDeBERTa | 0.824 | **0.812** | **0.801** | **0.791** | 0.822 | 0.813 | 0.802 | 0.797 | 0.803 | **0.795** | **0.772** | **0.755** |
| CF (Ours) | ST | 0.791 | 0.784 | 0.776 | 0.772 | 0.789 | 0.782 | 0.776 | 0.772 | 0.771 | 0.770 | 0.753 | 0.751 |
| CF (Ours) | mDeBERTa | 0.793 | 0.787 | 0.781 | 0.777 | 0.797 | 0.789 | 0.783 | 0.777 | 0.777 | 0.775 | 0.761 | 0.755 |

Table 11: FC-only and non-FC-only routing evaluation (mDeBERTa-CF). Routing within homogeneous capability groups.

| **Setting** | **Acc.** | **F1** | **Route Acc.** |
|---|---|---|---|
| FC-only (36) | 0.8794 | 0.9339 | 0.8633 |
| Non-FC-only (39) | 0.9303 | 0.9639 | 0.9330 |

Table 12: Same-family and same-size routing accuracy (mDeBERTa-CF). Models must select the best model within a homogeneous group.

| **Setting** | **# Models** | **Route Acc.** | **Pred Acc.** |
|---|---|---|---|
| Same family (GPT) | 8 | 0.807 | 0.826 |
| Same family (Claude) | 6 | 0.879 | 0.874 |
| Same family (Gemini) | 6 | 0.879 | 0.882 |
| Same family (Qwen) | 6 | 0.820 | 0.828 |
| Same family (Llama) | 6 | 0.759 | 0.786 |
| Same size ($\leq$3B) | 15 | 0.794 | 0.818 |
| Same size (4B–12B) | 14 | 0.944 | 0.941 |
| Same size ($\geq$20B) | 12 | 0.898 | 0.906 |

### C.3 Latent Space Visualization

To quantify which categorical distinction dominates the learned latent space, we compute the Adjusted Rand Index (ARI) between K-means cluster assignments and three ground-truth partitions of the model set.

In the full concatenated space, FC versus non-FC capability yields the highest ARI (0.136), exceeding model size (0.029) and provenance (0.007). The correctness sub-space shows weak clustering alignment across all partitions (ARI $\leq$ 0.030), while the latency sub-space is most aligned with model size (ARI = 0.077). The token sub-space shows near-random alignment with all partitions.

### C.4 Few-Shot Stability and Seed Variance

Table 14 reports few-shot adaptation results across 10 random seeds, showing mean and standard deviation for each training size.

The low variance across seeds indicates stable collaborative filtering-style cold-start adaptation.

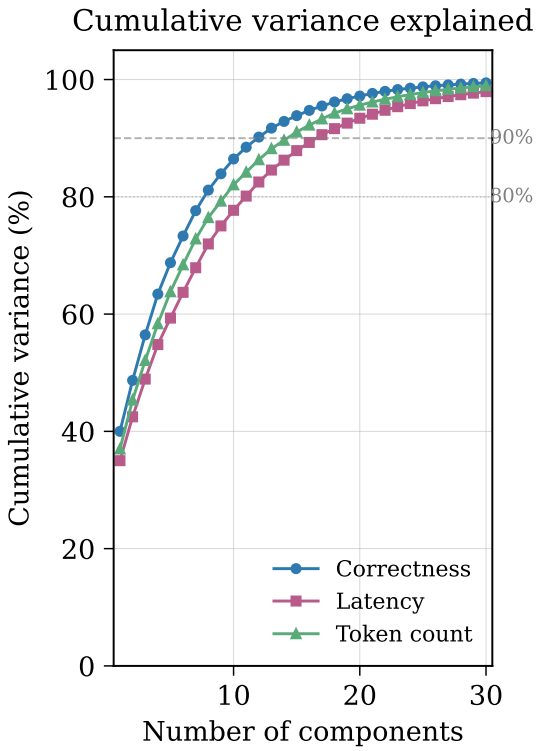



Figure 6: Cumulative variance explained. The top 10 components capture the majority of variance across all matrices.

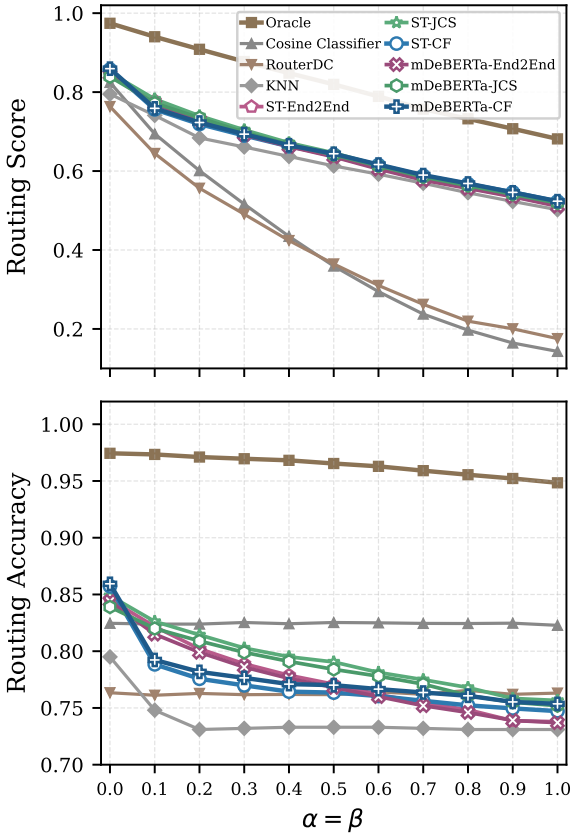



Figure 7: Routing score and accuracy as $\alpha = \beta$ varies. Our methods significantly outperform routing baselines across all cost weights.

Table 13: Clustering alignment (ARI) with ground-truth partitions.

| Partition | Full | Corr. | Lat. | Tok. |
|---|---|---|---|---|
| FC / Non-FC | 0.136 | 0.025 | 0.017 | −0.002 |
| Size (S/M/L) | 0.029 | 0.030 | 0.077 | −0.000 |
| Open / Proprietary | 0.007 | 0.028 | −0.026 | −0.025 |

Table 14: Few-shot adaptation stability (10-run mean ± std, mDeBERTa-CF).

| Train Size | Acc. | F1 | L-MAE↓ | T-MAE↓ |
|---|---|---|---|---|
| 100 | 0.778±.011 | 0.819±.012 | 0.082±.004 | 0.094±.004 |
| 250 | 0.798±.007 | 0.836±.008 | 0.076±.002 | 0.087±.002 |
| 500 | 0.804±.005 | 0.840±.005 | 0.075±.001 | 0.085±.002 |
| 1000 | 0.806±.006 | 0.841±.006 | 0.074±.001 | 0.084±.002 |