TensorOpera Router: A Multi-Model Router for Efficient LLM Inference

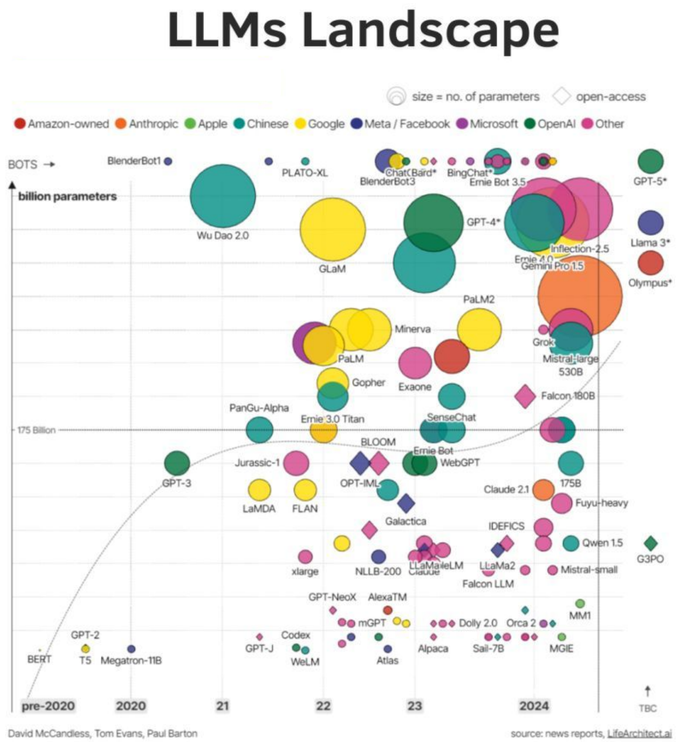
Dimitris Stripelis, Zijian Hu, Jipeng Zhang, Alay Dilipbhai Shah, Han Jin, Yuhang Yao, Jipeng Zhang, Tong Zhang, Salman Avestimehr, Chaoyang He

TensorOpera AI

Motivation

TensorOpera AI

- Different tasks require different expertise.
- Various and diverse LLMs have emerged.
- No LLM fits all purposes.

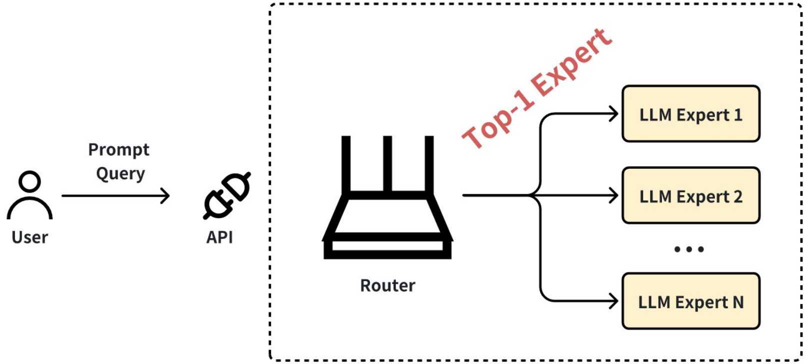


Problem Statement

TensorOpera AI

Given the increasing number and diversity of large language models available, there is a need for efficient and intelligent routing systems that:

- Dynamically route queries/tasks to the most suitable LLM expert.
- Optimize resource utilization and costs by balancing workload across models.
- Single API interface for multiple models.
- Seamless selection of expert models while abstracting complexity.

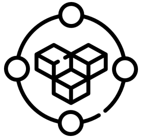


Challenges

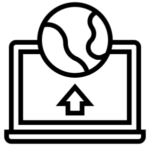
TensorOpera AI

- **Router model training and testing data preparation.**
  - Router's training and testing data need to be representative of the domain queries the router is expected to handle during deployment.

- **Router model selection.**
  - Router model needs to be lightweight to be easily deployable both on the edge and cloud, and quickly parse and assign queries to the most suitable expert.

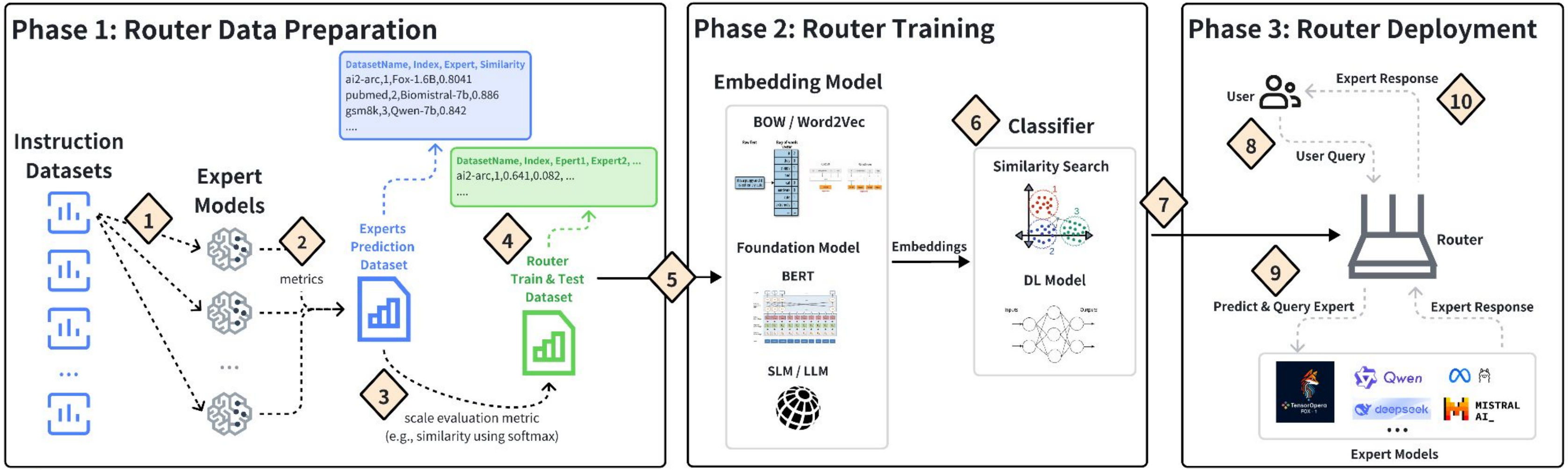


- **End-to-end routing service deployment.**
  - Automating the training and testing data generation, routing model selection and deployment in real-world setting is an extremely challenging engineering effort.



TensorOpera Router Approach

TensorOpera AI



1 Prompt expert models using instruction datasets.

2 Collect performance metrics and create expert dataset with all metrics.

3 Select metric for router training and testing and scale it to probabilities.

4 Prepare router's final training and testing datasets.

5 Pass instruction data through embedding model.

6 Fine-Tune / Train router's classifier on the embeddings.

7 Deploy trained router model.

8 Submit user query to router.

9 Router predicts expert to execute query.

10 Router replies expert's response to user.

Phase 1: Router Data Preparation

TensorOpera AI

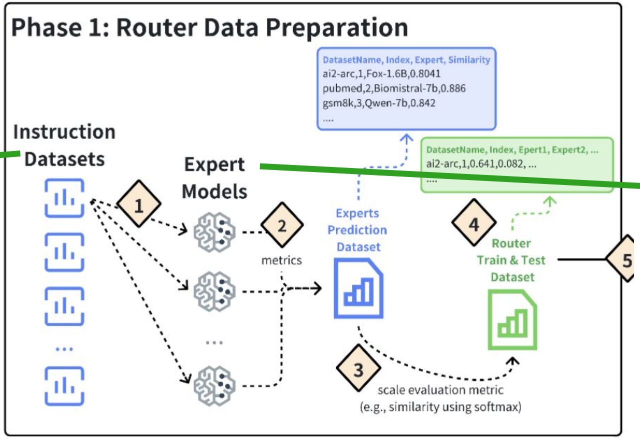
Instruction Datasets

Ai2-ARC (natural sciences)

GSM8k (math)

MBPP (code)

PubMedQA (medical)



Standalone LLM Experts

BioLlama-7B (medical)

BioMistral-7B (medical)

CodeLlama-7B (code)

Fox-1.6B (general)

Mistral-7B (general)

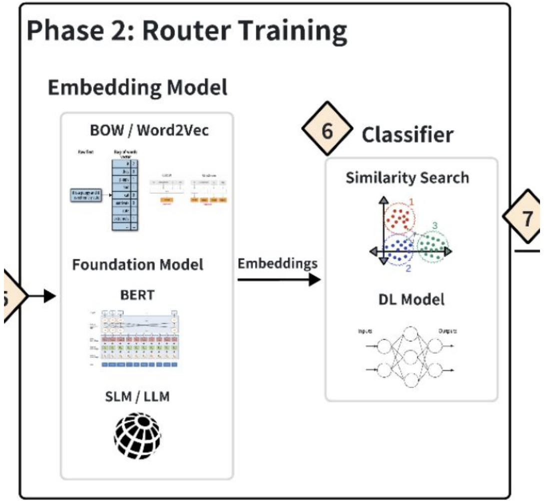
Qwen-7B (general)

MathDeepSeek-7B (math)

For every instruction prompt we collect:

- Negative Log Likelihood
- BERT embeddings similarity score (BERTSim)
- Inference Time (in seconds)
- Total Input Tokens
- Total Output Tokens

Phase 2: Router Training



5 Pass instruction data through embedding model.

6 Fine-Tune / Train router's classifier on the embeddings.

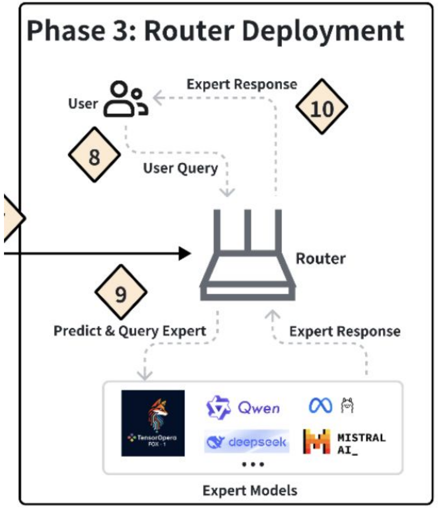
Original Data: (Instruction, BERTSim1, …, BERTSimN)

Scaled Data w/ Softmax Labels:
(Instruction, softmax1(BERTSims, …, BERTSimN), …, softmaxN(BERTSim1, …, BERTSimN))

- **MLP-Router**
  - convert all training queries into their vector representation by fitting a Bag-of-Words model, use cross-entropy loss on the scaled BERTSim scores.
- **BERT-Router**
  - add a classifier head on base BERT and fine-tune BERT by training on the BERT embeddings of all training queries, using cross-entropy loss on the scaled BERTSim scores.

Phase 3: Deployment

TensorOpera AI



7 Deploy trained router model.

8 Submit user query to router.

9 Router predicts expert to execute query.

10 Router replies expert's response to user.

Router Endpoint Query

```
curl -XPOST http://0.0.0.0:2345/predict -H 'Accept: application/json' -H 'Content-Type:
application/json' -d '{ "messages": [{ "role": "user", "content": "Test" }] }'
```

PolyRouter Service

docker

Router Endpoint Reply


```
{"id":"d4961180721145b1916c7c18e935db6f","object":"chat.completion","created":4318860,"
choices":[{"index":0,"message":{"role":"assistant","content":"It looks like you're
testing me! Is there something specific you'd like to know or discuss? I'm here to
help!"},"finish_reason":"stop"}],"usage":{"prompt_tokens":29,"total_tokens":56,"complet
ion_tokens":27},"expert":"llama3-8b-cloud"}
```

Evaluation Criteria & Baseline Methods

TensorOpera AI

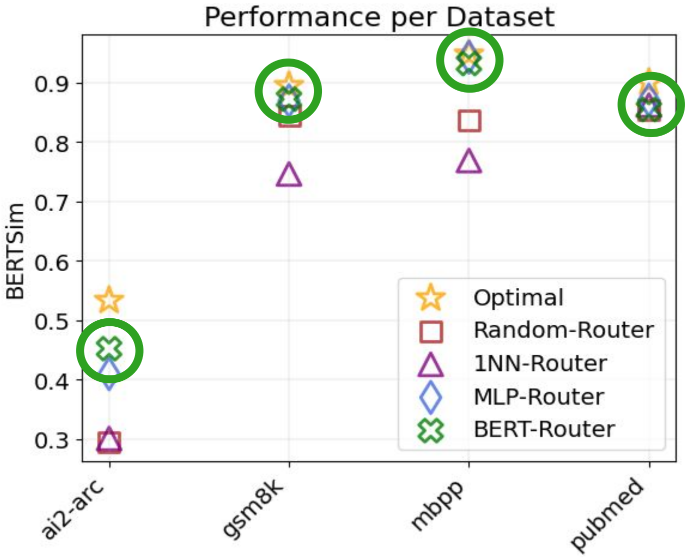
Criteria

- Inference Price Cost ($$/1M tokens)
- Throughput (#tokens/sec)
- BERT Similarity (cosine similarity on BERT embeddings)
- Negative Log-Likelihood (NLL)

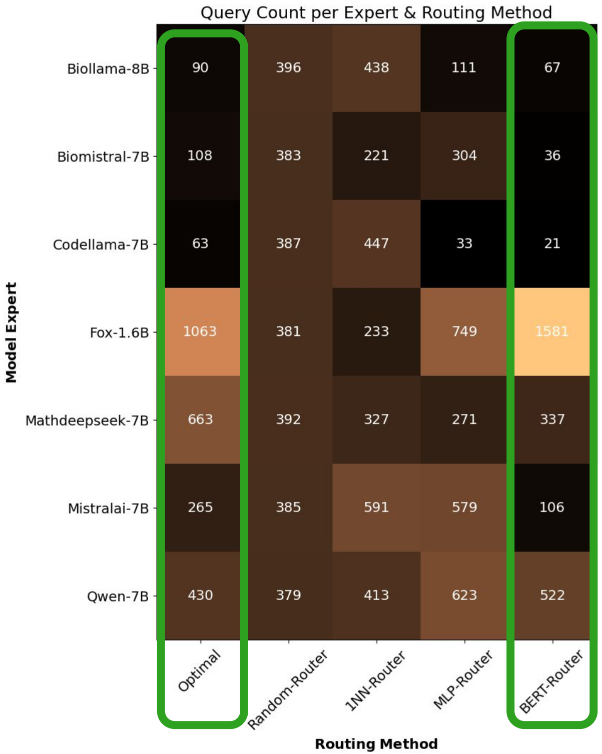
Routing Baselines

- **Zero-Router**
  - average performance of all available experts without any routing logic
- **Optimal-Router**
  - for any given query the optimal set of values is the minimum cost, maximum throughput, maximum BERTSim, minimum NLL recorded by any expert model or routing method.
- **Random-Router**
  - pick a random expert from all available experts.
- **1NN-Router**
  - for every test query find its closest training query (w.r.t. embedding space) and assign the expert that exhibited the best performance for the training query.

Router Performance per Dataset & Query Cardinality per Expert



BERT Router exhibits the best performance



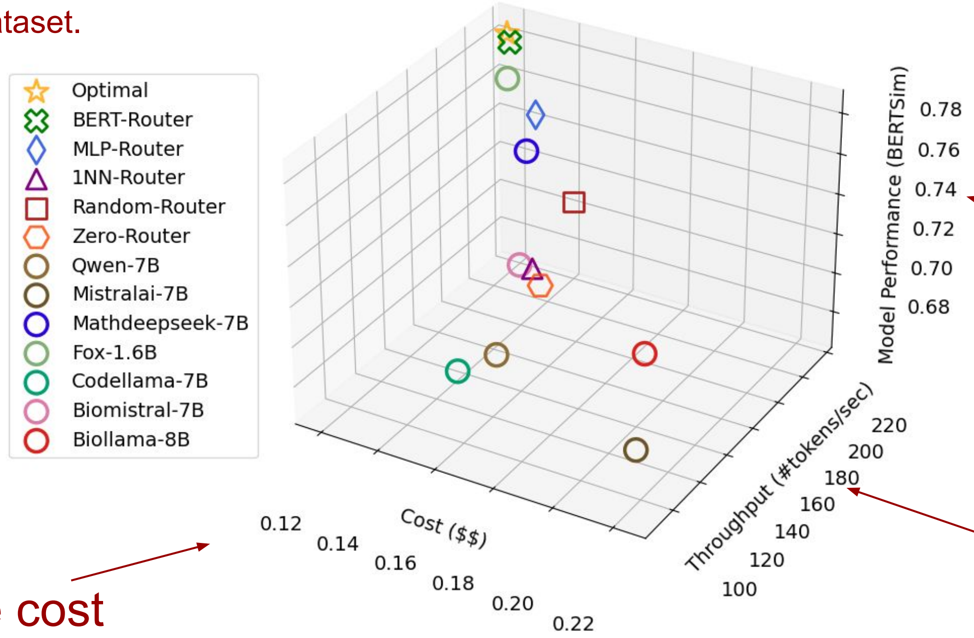
Model Expert	Optimal	Random-Router	1NN-Router	MLP-Router	BERT-Router
Biollama-8B	90	396	438	111	67
Biomistral-7B	108	383	221	304	36
Codellama-7B	63	387	447	33	21
Fox-1.6B	1063	381	233	749	1581
Mathdeepseek-7B	663	392	327	271	337
Mistralai-7B	265	385	591	579	106
Qwen-7B	430	379	413	623	522

BERT Router assigns similar number of queries per expert as Optimal

Trilemma Evaluation (Cost, Throughput, Performance)

TensorOpera AI

Test Queries: random 20% selection from each benchmark dataset.



avg performance
(across all test queries)

avg throughput
(across all test queries)

avg price cost
(across all test queries)

Future Directions
Hybrid AI
Edge-to-Cloud Collaboration