Aneesh Bendale, Lennard Korte, Joann Wong

Project Exposé

LLM Router Model Comparison with Preference Data

22th october 2024

OVERVIEW

We address the challenge of balancing performance and cost in large language models (LLMs) by developing a router that dynamically selects between stronger, expensive models and weaker, cost-effective ones. Building on the work of routing between LLMs by RouteLLM (2024), we extend existing routers with our own approaches to further improve cost and performance for our models. Our new system routes queries across a larger set of models, rather than being limited to two, allowing for even higher performance while maintaining cost efficiency. Additionally, we compare and test the routers with our newly developed models to validate improvements.

OBJECTIVES

One of the key aspects of RouteLLM is how it predicts the best model for a given query based on past performance. Their proposed framework uses methods like **Matrix Factorization, BERT Router, Causal LLM Router**, and **Similarity-Weighted Ranking.** To improve this, we propose the following -

1. **Using existing and new models to route between larger LLM sets:**
    a. i.e. 3-4 instead of differentiating between strong and week only
    b. sorted by strength
    c. sorted by category (i.e. maths, politics, etc.) and mixed -> not sure yet, cuz this would require accurate benchmarks?)
2. **Matrix factorization extended with neural network layers**
3. **Different Feature Engineering:**

    We aim to train our router architectures based on various features, among which we introduce newly engineered features. A critical factor when generating features is the

computational cost. Thus, we differentiate between features that are generated using model-based and model-less features

a. **Model-less feature generation:**
    i. Using basic computations
    ii. Or using database lookups: i.e.: prompt length, prompt theme category, lexical ambiguity, redundancy, etc. (i.e. compare keywords to DB)
b. **Model-based feature generation:**
    i. Test different model complexities
    ii. Using transformers, traditional NLP techniques and shallow neural networks, options:
        1. **Extract interpretable feature:** i.e. topic categorization / domain classification (BERT-classifier), complexity, Syntactic complexity (SpaCy) or other prompt properties (semantic evaluation of a prompt) to be able to weigh these with additional models
        2. **Extract uninterpretable features:** (embeddings using Sentence-BERT or MiniLM to capture semantic meaning): as input for other architectures

4. **Decision Layer: Develop new architectures / architecture variations and compare with existing router architecture performances:**
    a. Prompt as input using NLP models with different model complexities as router directly
    b. Architectures utilizing the generated features using a traditional multi-class classifier:
        i. MLP
        ii. Logistic Regression
        iii. Random Forest
        iv. Decision tree
        v. SVM or Gradient Boosting
5. **Ensemble Models (which ones TBD) to improve the routing decision-making process.**
6. **Possible two step router**
    a. **Example workflow:**
        i. **Easy queries** are immediately routed to the weak model.
        ii. **Difficult queries** are routed to the strong model.
        iii. **Moderate queries** are passed to the next step for further analysis.
    b. **Moderate Query Router**

        i. **Decision tree or SVM (why these?)** to determine whether the query should be routed to the middle-tier model or escalated to the strong model.

7. **New Architectures:**
    a. **Complexity scoring layer**
        i. Based feature extraction, calculate an aggregated complexity score for the query
    b. **Dynamic cost threshold layer**
        i. A dynamic threshold adjusts in real-time based on the complexity score of each query, making the routing more responsive to variations in query difficulty. This allows for more fine-grained control over when to use the strong model, improving cost-efficiency and maintaining high quality for complex queries

Archive:

c. **Decision Layer-**

i. Initial Query Complexity Classifier -

- Using a traditional multi-class classifier (like **Logistic Regression**, **Random Forest**, **Decision tree, SVM** or **Gradient Boosting**) to classify queries into three categories: **easy**, **moderate**, or **difficult**.
- **Workflow**:
    - **Easy queries** are immediately routed to the **weak model**.
    - **Difficult queries** are routed to the **strong model**.
    - **Moderate queries** are passed to the next step for further analysis.

ii. Moderate Query Router -

- **Decision tree** or **SVM** to determine whether the query should be routed to the **middle-tier model** or escalated to the **strong model**.