

AI Prompt Studio: Domain-Aware Model Selection Using Adaptive Learning

Shivam Ramdas Jagtap, Siddhi Vijay Khandave, Soham Sanjay Khandve, Ritesh Pravin Shinde, Prof. Kiran Gaikwad

[1]Student [2]Student [3]Student [4]Student [5]Professor

Dept. of Artificial Intelligence and Data Science

Marathwada Mitra Mandal's Institute of Technology, Lohegaon, Pune

Abstract —The rapid expansion of Large Language Models (LLMs) has introduced a new challenge: selecting the right model for a given task without exhausting computational resources or incurring unnecessary API costs. AI Prompt Studio addresses this by introducing a hybrid domain-aware model selection system backed by persistent MongoDB memory. Instead of querying multiple models in parallel for every prompt, the system first checks whether a domain-to-model mapping already exists in its memory. If found, the best-suited model for that domain is called directly. If not, the user manually selects the domain through a clean frontend interface, and that mapping is stored for all future prompts. Responses are evaluated by GPT-4 across three quality dimensions — Relevance, Creativity, and Factual Accuracy — returning a structured JSON score. Built with FastAPI on the backend and React.js on the frontend, the system demonstrates measurable reductions in API usage and response latency while maintaining and even improving output quality. This work presents an efficient, adaptive, and practically deployable approach to intelligent prompt routing for LLM-powered applications.

Keywords: Prompt Engineering, Domain-Aware Routing, Adaptive Learning, MongoDB, FastAPI, LLM Evaluation, GPT-4, Groq API, Hybrid Intelligence

I. INTRODUCTION

The emergence of Large Language Models has created a new class of engineering challenge: not just how to use these models, but which model to use for which task. Every LLM excels in a specific context — some are strong in creative tasks, others perform better on factual or technical queries. Without a principled approach to model selection, developers either default to the most powerful (and expensive) model for everything, or run all available models in parallel and pick the best response after the fact. Both strategies waste resources.

AI Prompt Studio was originally developed as a unified prompt engineering platform where every prompt was sent to multiple models — ChatGPT, Gemini, and Claude — and responses were evaluated for Relevance, Creativity, and Factual Accuracy before presenting the best result. While this multi-model approach established a solid quality baseline, it carried a fundamental inefficiency: for every single user query, three API calls were made, tripling cost and compounding latency.

The Semester 2 implementation resolves this by introducing a hybrid domain learning strategy. The core insight is simple — a user's domain of interest tends to be consistent. A medical professional asking health questions today is likely to ask similar questions tomorrow. If the system learns which model performs best for a given domain, it can route directly to that model without re-evaluating alternatives every time.

The proposed system maintains a domain-to-model mapping in MongoDB. On each request, the FastAPI backend performs a fast indexed lookup. If the domain is recognized, a single API call is made to the best model for that domain. If not, the user selects the domain from a dropdown in the frontend, the system stores the mapping via the /save-domain endpoint, and all subsequent prompts in that domain are handled automatically. GPT-4 evaluates every response and the interaction history is persisted for future reference. This architecture delivers a 66% reduction in API calls, approximately 67% faster response times, and marginally improved evaluation scores compared to the original multi-model pipeline.

II. LITERATURE REVIEW

From Manual to Automated Prompt Engineering: Evolving LLM Prompts with Genetic Algorithms [1]

Loss and Dhuvad demonstrated that Genetic Algorithms can systematically evolve LLM prompts, achieving an average 24% accuracy improvement over manually crafted prompts across four domain-specific tasks including academic topic classification, financial math reasoning, and sentiment analysis. Their work treats candidate prompts as individuals in a population, applying selection, crossover, and mutation operators to evolve increasingly effective prompt strategies. The study validated across GPT-4o, GPT-4o-mini, Mistral NeMo, and Llama 3.2, proving that automated prompt optimization consistently outperforms human prompt engineers even on unfamiliar domains. The core insight — that domain-specific strategies emerge naturally from systematic optimization — directly motivates our approach of learning the domain-model relationship once and reusing it rather than re-evaluating on every query.

Optimizing Human-AI Interaction: Innovations in Prompt Engineering [2]

Deshmukh et al. proposed a structured methodology for prompt optimization that emphasizes clarity, specificity, and contextual grounding. Their architecture combines a basic prompt input with a set of user preferences (role, tone, length) fed into a fine-tuned GPT-3.5 model, producing enhanced prompts evaluated against a reward function based on user feedback. The study introduced a synthetic dataset of 50,000 prompt pairs and demonstrated that iterative, feedback-driven prompt refinement significantly improves response quality across domains including healthcare and education. The principle that structured, domain-aware interaction between user and model yields better outcomes than generic prompting underpins our domain classification and evaluation pipeline.

A Survey of Automatic Prompt Engineering: An Optimization Perspective [3]

Li et al. presented the first unified optimization-theoretic framework for automated prompt engineering across discrete, continuous, and hybrid prompt spaces. The survey categorizes methods into FM-based optimization, evolutionary computing, gradient-based optimization, and reinforcement learning paradigms. Crucially, the authors identify domain-specific task optimization as an underexplored frontier and highlight the need for systems that balance interpretability, cost, and accuracy. Their formulation — selecting the optimal prompt P^* as the argmax of expected performance over a validation set — directly informs our mathematical model selection criterion. The survey's emphasis on reducing human intervention while preserving domain alignment is precisely what our adaptive memory mechanism achieves.

III. METHODOLOGY

The proposed AI Prompt Studio system is developed using a structured, modular approach that separates domain learning, model execution, and quality evaluation into distinct, independently testable components. The methodology consists of five stages: domain lookup, model selection, API execution, GPT-4 evaluation, and persistent storage.

In the first stage, every incoming prompt is accompanied by a domain identifier. The FastAPI backend performs an indexed MongoDB query on the `domain_mappings` collection. This lookup typically completes in under 10ms and returns either a stored model name or a null result. This binary outcome determines the entire execution path for that request.

In the second stage, the `ModelSelector` component implements the domain-aware argmax strategy formalized as: $\text{Model_selected} = \text{argmax}(M_i \in \text{Models}) \text{Performance}(\text{Domain}, M_i)$. For known domains, the stored best model is retrieved directly. For unknown domains, the frontend renders a domain selection dropdown, the user's choice is transmitted to the `/save-domain` endpoint, and the mapping is written to MongoDB before execution proceeds.

In the third stage, only one API call is made — to the selected model via either the Groq API client (for LLaMA-based models) or the OpenAI client (for GPT models). All API calls are wrapped in timeout-aware handlers to prevent blocking. In the fourth stage, the `EvaluationEngine` sends the original prompt and model response to GPT-4 with a structured system prompt requesting scores for Relevance, Creativity, and Factual Accuracy on a 0–10 scale, returned as a JSON object with an average composite score. In the fifth stage, the complete interaction record — prompt, domain, selected model, response, scores, and timestamp — is written to the MongoDB `prompt_history` collection and returned to the frontend for display.

IV. IMPLEMENTATION

The implementation of AI Prompt Studio is carried out in a structured step-by-step manner using React.js, FastAPI, MongoDB, and the Groq/OpenAI APIs to ensure modularity, efficiency, and scalability.

Step 1: System Setup The development environment is configured by installing Python, Node.js, FastAPI, and MongoDB. Required libraries including pymongo, openai, groq, uvicorn, and React dependencies are installed via pip and npm respectively.

Step 2: Frontend Development (React.js) The user interface is developed using React.js with components for prompt input, domain selection dropdown, response display, and evaluation score visualization. Axios handles all API communication with the FastAPI backend.

Step 3: Backend Development (FastAPI) The backend is built with FastAPI, exposing four REST endpoints: POST /prompt for prompt processing, GET /domain-check for domain lookup, POST /save-domain for persisting new domain-model mappings, and GET /history for retrieving conversation records.

Step 4: Database Integration (MongoDB) MongoDB is used with two collections: domain_mappings (indexed on domain key for fast lookup) and prompt_history (stores full interaction records with timestamps). PyMongo with Mongoose-equivalent schemas manages all database operations.

Step 5: Domain Lookup Module On receiving a prompt, the backend queries the domain_mappings collection. If a mapping exists, the stored model name is returned immediately. If not, a domain_required flag is sent to the frontend, triggering the domain selection UI.

Step 6: Domain Save Module When the user selects a domain, the frontend sends a POST request to /save-domain with the domain name and the identified best model. This mapping is stored in MongoDB and all future prompts with that domain are automatically routed without user input.

Step 7: Model Execution Only one model is called per prompt. The GroqAPIClient or OpenAIClient is instantiated based on the selected model and calls the completion endpoint with the user's prompt. Timeout handling prevents blocking on slow API responses.

Step 8: GPT-4 Evaluation Engine The EvaluationEngine constructs a structured evaluation prompt instructing GPT-4 to return a JSON object with keys relevance, creativity, factual_accuracy (each 0–10), and avg. Python's json.loads with regex fallback handles response parsing.

Step 9: History Storage The complete interaction — prompt, domain, model used, response text, evaluation scores, and timestamp — is written to the prompt_history collection. This enables session persistence and future analytics.

Step 10: Testing and Debugging Each module is unit tested independently and then integration tested end-to-end. API response correctness, domain routing logic, and evaluation JSON parsing are verified before deployment.

Step 11: Deployment The FastAPI backend is deployed using Uvicorn and the React frontend is built for production. Environment variables for OPENAI_API_KEY and GROQ_API_KEY are configured securely outside the codebase.

V. RESULT AND DISCUSSION

The AI Prompt Studio system demonstrated effective performance in adaptive model routing and quality-assured response generation. Comparative testing was conducted across 60 prompts spanning six domains: Medical, Legal, Technical (Code), Creative Writing, Educational, and General. The Semester 2 domain-memory pipeline was compared against the original multi-model execution approach after domain mappings had been established for all domains.

The system showed high responsiveness in domain lookup with average MongoDB query times under 10ms. Single-model execution reduced average response time from approximately 4.2 seconds (multi-model parallel execution) to approximately 1.4 seconds — a 67% improvement. API call volume dropped from 3 calls per prompt to 1, reducing cost proportionally. GPT-4 evaluation scores improved marginally across all three dimensions, indicating that targeted domain-specific routing produces responses that are more contextually aligned than a generalist model selected from parallel outputs.

Metric	Sem 1 (Multi-Model)	Sem 2 (Domain Memory)	Improvement
Avg. API Calls/Prompt	3	1	66% reduction
Avg. Response Time	~4.2 sec	~1.4 sec	~67% faster
Avg. Relevance Score	7.8 / 10	8.1 / 10	+0.3

Avg. Creativity Score	7.5 / 10	7.7 / 10	+0.2
Factual Accuracy Score	8.0 / 10	8.3 / 10	+0.3
API Cost (relative)	3× baseline	1× baseline	3× reduction

System Performance Analysis

After the first prompt in each domain, all subsequent prompts were routed without user intervention, confirming the adaptive learning behavior described in the methodology. The domain_mappings collection grew naturally across sessions with no manual maintenance. MongoDB's indexed lookup ensured that the domain check added negligible overhead to the total request pipeline.

VI. TESTING

Step 1: Unit Testing Each individual module — domain lookup, model client, evaluation engine, and history writer — is tested in isolation to verify correct behavior. Edge cases such as missing domain keys, API timeouts, and malformed JSON evaluation responses are handled and verified.

Step 2: Integration Testing All modules are combined and tested end-to-end to verify correct data flow between the React.js frontend, FastAPI backend, and MongoDB database. API request-response cycles for all four endpoints are validated for correct status codes and payload structure.

Step 3: System Testing The complete application is tested as a whole across all six supported domains. Both the known-domain fast path and the unknown-domain user-selection path are exercised. GPT-4 evaluation correctness and JSON parsing resilience are verified under real API conditions.

Step 4: User Acceptance Testing (UAT) The system is tested by end users to evaluate usability of the domain selection interface, clarity of evaluation score display, and overall satisfaction with response quality. Feedback confirmed that the domain selection step (for first-time domains) was intuitive and non-disruptive.

Step 5: Performance and Security Testing Response time, API timeout handling, and MongoDB query performance are measured under concurrent load. Environment variable-based API key management and no key exposure in frontend bundles are verified. MongoDB collections are access-controlled to prevent unauthorized reads or writes.

VII. FUTURE SCOPE

The current domain-aware routing system can be extended in several meaningful directions. In the near term, automatic domain classification using a lightweight text classifier — such as a fine-tuned BERT model or zero-shot classifier — would eliminate the manual labeling step entirely for first-time domains, making the system fully autonomous from the very first prompt.

The static domain-model mapping can be replaced with a dynamic scoring system that periodically re-evaluates model performance per domain as new responses accumulate, automatically updating mappings when a previously inferior model begins outperforming the currently stored choice. This would allow the system to adapt to ongoing model updates from providers.

Integration with Retrieval-Augmented Generation (RAG) pipelines would allow domain-specific document stores — medical literature, legal databases, technical manuals — to augment model responses with verified source material, significantly improving factual accuracy for high-stakes domains. Additionally, multi-modal prompt support would extend routing to image-plus-text queries, leveraging vision-capable models for domains that require visual analysis. With continued improvements in scalability and domain coverage, AI Prompt Studio has the potential to serve as a comprehensive intelligent prompt routing layer for LLM-powered applications across enterprise and public-sector deployments.

VIII. CONCLUSION

AI Prompt Studio Semester 2 delivers a concrete and measurable improvement over its predecessor by replacing expensive multi-model parallel execution with a domain-aware single-model routing strategy backed by persistent MongoDB memory. The hybrid learning approach — combining user-guided domain initialization for first-time domains with fully automatic memory-based routing for all subsequent prompts — achieves a 66% reduction in API

calls, approximately 67% reduction in response latency, and marginal improvement in GPT-4 evaluation scores across Relevance, Creativity, and Factual Accuracy dimensions.

The system demonstrates that domain awareness, implemented through a simple key-value memory store and a lightweight routing decision in the FastAPI backend, is sufficient to deliver substantial engineering value without requiring complex machine learning infrastructure. The modular architecture supports straightforward addition of new models or domains. The core principle established by this work — learn the domain-model relationship once, route intelligently thereafter — represents a broadly applicable design pattern for cost-effective, quality-preserving LLM system design in real-world deployments.

VI. REFERENCES

- [1] L. A. Loss and P. Dhuvad, "From Manual to Automated Prompt Engineering: Evolving LLM Prompts with Genetic Algorithms," 2025 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8, 2025. DOI: 10.1109/CEC65147.2025.11043059
- [2] R. Deshmukh, S. Gurav, R. Raut, Y. Patil, and M. Bhavsar, "Optimizing Human-AI Interaction: Innovations in Prompt Engineering," in Proc. 3rd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT-2025), IEEE, pp. 1240–1246, 2025. DOI: 10.1109/IDCIOT64235.2025.10914815
- [3] W. Li, X. Wang, W. Li, and B. Jin, "A Survey of Automatic Prompt Engineering: An Optimization Perspective," arXiv preprint arXiv:2502.11560, 2025.