Provisional Patent Application: UltrLLMOrchestrator System

01. **Title of the Invention:** UltrLLMOrchestratorSystem, Method, and Class for Advanced Multi-Modal Artificial Intelligence Symbiosis (the "invention").

02. **Field of The Invention:** This invention relates to the field of artificial intelligence (AI) and natural language processing (NLP). Specifically, it concerns programs, systems, and methods for orchestrating interaction among multiple AI models to generate enhanced intelligent text-based processing, analyses, synthesis, and output. The invention further addresses the user-centric experience and interaction with AI, seeking to maximize the efficacy of the human/AI interaction such that the user receives outputs that are in line with their reason for using AI in the first place.This invention relates to artificial intelligence systems, specifically addressing the orchestration of multiple large language models. The system enables sophisticated coordination between different AI models through asynchronous processing and multi-stage refinement, creating a unified framework for enhanced analysis capabilities.

03. **Background of the Invention:** Artificial Intelligence emerged in 1956, but it didn't breakthrough until the 20100s with enabled by increased computing power and large datasets. By the early 2020s, large language models and foundation models demonstrated unprecedented capabilities in understanding and generating human-like text, code, and multimedia content, marking the beginning of a new era in artificial intelligence with broad implications across industries.

    By 2024, several companies had developed highly capable Large Language Models (LLMs) that served as consumer-facing products. To access the LLMs, users typically purchased access with monthly subscriptions and interacted with each individual LLM via its official user interface. Each of these LLMs were "trained" using different data and different training techniques,. But all emplo a deep neural network program.

    utputs even when given identical prompt inputs. using different models is These models lacked sophisticated orchestration capabilities that utilize multiple AI models to process, analyze, and react to user inputs in a collaborative and amplifying manner. This limited the depth and breadth of multi-modal AI applications.

    urrent approaches to AI model integration face significant limitations. Traditional implementations typically process requests through single large language models or employ basic sequential combinations of models. This creates two fundamental problems: inefficient resource utilization and failure to leverage potential synergies between models.Existing solutions attempt to combine multiple models through simple aggregation or voting mechanisms. However, these approaches lack sophisticated mechanisms for cross-model refinement and synthesis. The sequential processing of model responses leads to increased latency and reduced efficiency. Additionally, current systems fail to address the complex challenge of managing varying model capabilities, response times, and error conditions in a coordinated manner.

04.
    a. The invention was created to leverage the collective intelligence that multiple AI models could bring to address user inquiries, commands, and challenges in real-time, applying individual AI models in unison to produce an output influenced by all, leveraging their unique strengths to collectively produce outputs that are better than those they would produce alone.
    b. The tangible elements of the invention is a set of code using python coding language that directs a computer to activate the individual LLMs while balancing the limitations of the computer and restrictions of LLM access to produce accurate, consistent and well rounded outputs that satisfy the user's requirements and reasons for using the technology.
    c. To successfully apply the collective intelligence of multiple LLMs , the invention had Central challenges to balancing the needs of an effective multi-modal t multimodal applications face challenges in:
        i. Efficient coordination of multiple AI models
        ii. Rate limiting
        iii. Dynamic resource allocation
        iv. Real-time performance optimization
        v. Robust error handling and recovery
        vi. Adaptive learning from system behavior
        vii. Efficient Unified response creation
    d. The UltrLLMOrchestrator System addresses these challenges with a flexible, scalable, and effective architecture that utilizes the best traits of premium LLMs in concert with one another, effectively multiplying the intelligence applied to solve a problem. It
        i. provides a comprehensive and nuanced understanding of input data,
        ii. Employs effective techniques to work to address broad problems and discrete tasks collectively and
        iii. produces outputs that

1. feature exhaustive, yet never superfluous depth
2. Are factually accurate and logically consistent throughout and
3. Are rational -- contextually well-rounded and toned consistent with user's expectations

05. **Description/Summary of the Invention**: The UltrLLMOrchestratoris an AI orchestration system designed to integrate and manage the workflow of multiple AI generative models to collectively perform analysis, meta-analysis, and synthesis of data or textual prompts in an automated, cohesive process. This orchestration allows the multimodal system to perform more comprehensive and nuanced analyses and more accurate outputs than individual models. The invention represents a novel approach to orchestrating multiple Large Language Models (LLMs) through an innovative multi-layer architecture that enables dynamic model selection, intelligent resource management, and adaptive optimization. The system uniquely combines multi-model orchestration, multi-layered analysis, resource management, and error handling to create a robust, efficient, and scalable solution for AI model management and execution. This system introduces an innovative approach to initial response generation, meta-analysis, and synthesis across models, culminating in a hyper-level synthesis that draws unique insights and perspectives by combining the outputs in a structured manner.

    The present invention introduces a novel system for orchestrating multiple large language models through a sophisticated three-phase process. At its core, the system implements asynchronous task distribution, allowing parallel model execution while maintaining coordinated output synthesis. This is achieved through a combination of template-based prompt management, exponential backoff retry mechanisms, and comprehensive logging systems.
06. The invention's primary innovation lies in its ability to enable dynamic interaction between multiple AI models, creating a coordinated system that exceeds the capabilities of its individual components. This is accomplished through a structured yet flexible architecture that adapts to varying model capabilities and task requirements.

07. **Detailed description of key system**s

    a. Key Systems:

        i. **Dynamic Multi-Model Orchestration:** Innovation: Leveraging Unique Strengths: The system aims to harness the unique attributes of each AI model to perform individual and collective analyses on prompts.Introduces a unique model registry and dynamic selection system that adapts to real-time performance metrics and resource availability. Key Novel Components include:
            1. Adaptive model priority system based on historical performance
            2. Real-time model capability matching with request requirements
            3. Dynamic model loading and unloading based on demand patterns
            4. Unified interface abstraction across different model types
            5. Asynchronous Processing: It employs asynchronous processing techniques for parallel execution of API calls and orchestration tasks, significantly improving efficiency.
            6. Dynamic Selection of AI Models: The system can dynamically select AI models based on configured priorities or specific criteria.
            7. Flexible Output Formatting: The system offers configurable output formats to suit different applications and user preferences. It supports various response formatting options (e.g., plain text, markdown, JSON).

        ii. **Multi-Layered Analysis Pipeline:** Innovation: A key feature is its unique, multi-layered processing methodology that refines and synthesizes responses through several stages. This includes initial response generation, meta-analysis, ultra-synthesis, and hyper-level analysis.Implements a four-stage processing pipeline that enables progressive refinement and cross-model synthesis. Key Novel Components Keyword Extraction: Automatically extracts keywords from prompts.Multi-Layered Processing: Uses a multi-layered processing workflow to refine and synthesize response include:
            1. Meta-analysis phase for cross-model consistency checking
            2. Ultra synthesis stage for advanced response combination
            3. Hyper-level analysis for quality assurance
            4. Adaptive feedback loops for continuous improvement\
            5. Customizable Prompt Templates: The system uses unique prompt templates to guide AI model responses, ensuring consistent and efficient task execution. These templates facilitate tailored responses.
            6. Comprehensive Synthesis: The system enables a novel workflow that can be applied to various domains, from content creation and summarization to complex problem-solving and innovation brainstorming. The orchestration culminates in synthesized responses that reflect a multi-model perspective.

iii. **Intelligent Resource Management:** The system features dynamic management of API rate limits, hardware resources, and task scheduling. This ensures efficient operation across varying network conditions and API constraints.Innovation: Introduces an adaptive resource management system with predictive optimization.Resource Management and OptimizationResource management forms a critical component, with dynamic handling of API rate limits and intelligent hardware acceleration detection, including specialized support for architectures like Apple Silicon MPS. The system continuously monitors performance metrics and adapts resource allocation in Hardware Acceleration: Detects and leverages hardware acceleration (e.g., Apple Silicon GPU) for optimized performance.

1. Output Formatting: Formats responses from AI models according to a configurable output format.
2. Programming Languages: Can be implemented in Python, JavaScript, or C++.
3. Key Python Libraries: Utilizes libraries such as asyncio, requests, tenacity, and dotenv.
4. real-time, while maintaining thread-safe operations throughout its execution. Key novel components include:
5. Dynamic rate limiting with automatic adjustment: Integration of Rate Limiting and Retry Logic: Rate limiting and retry logic are integrated to ensure reliable operation across varying network conditions and API constraints. It employs retry mechanisms with exponential backoff to handle request failures gracefully.
6. Dynamic Adaptation: The orchestrator dynamically manages API rate limits, hardware resources, and can integrate both cloud-based and local AI services within a unified framework.
7. Hardware-aware resource allocation
8. Predictive scaling based on usage patterns
9. Real-time performance optimization

iv. **Advanced Error Handling & Reliability:** Innovation: Implements a comprehensive error management system with predictive and reactive components.Reliability is ensured through a comprehensive error handling system that implements exponential backoff retry mechanisms and adapts to varying network conditions. This is complemented by sophisticated task distribution that routes requests based on priority, context, and model capabilities, intelligently assigning models to specific stages of analysis based on their strengths.Dynamic Adaptation and Processing Methodology

1. A key innovation lies in the system's dynamic adaptation capabilities, continuously adjusting to changing conditions across hardware, API usage patterns, and network environments. The processing methodology implements a hierarchical synthesis approach that integrates analyses across models, producing more nuanced and comprehensive results than individual models could achieve alone.Template System and Integration FrameworkSelf-Analysis and Documentation Capabilities
2. Perhaps most uniquely, the system incorporates self-analysis capabilities, enabling it to document its own functionality, monitor its performance, and even assist in generating patent documentation. This introspective ability sets it apart from traditional orchestration systems, creating a more adaptive and self-aware processing environment.
3. The template system generates and adapts prompts dynamically based on context and processing stage. This works in concert with a flexible integration framework that unifies cloud and local services under a single coherent interface, ensuring consistent and efficient operation across diverse environments. Unique Capabilities include:
    a. Predictive error prevention
    b. Multi-stage recovery mechanisms
    c. Transaction-based state management
    d. Intelligent retry strategies

v. Find a place for these??

1. Dynamic AI Model Selection: Selects AI models based on configured priorities or task requirements.
2. Self-Analysis: Performs self-analysis for documentation and patent application purposes.
3. Open to application and adtaptation of flexible UXUI design

b. **Core Architecture**

i. Model Layer

1. ModelRegistry
2. ├── Dynamic Model Loading
3. ├── Capability Management
4. └── API

ii. User Layer
    1. User Profile/Preferences
    2. ├── Personalized UX/UI
    3. ├── Previous Sessions/User Learning
    4. └── Data transmission to Orchastrator for context/limits ce Abstraction
iii. Orchestration Layer
    1. ProcessingPipeline
    2. ├── User Input; Pretreatment
    3. ├── Orchestration Type/Applicaiton
    4. ├── Output
    5. └── Applied session tweaks; Reruns
iv. Resource Layer
    1. ResourceManager
    2. ├── Rate Limiting
    3. ├── Hardware Acceleration
    4. ├── Queue Management
    5. └── Performance Monitoring
v. Reliability Layer
    1. ErrorHandler
    2. ├── Retry Mechanisms
    3. ├── State Management
    4. ├── Recovery Strategies
    5. └── Error Reporting

c. **Data Flow:** Data Flow Diagram for UltrLLMOrchestrator

i. User Input
    1. A user provides an initial prompt to the system.
    2. The system analyzes this prompt to extract key concepts.
ii. Prompt Template and Management:
    1. Custom data classes and templates manage different levels of analysis prompts.
    2. Prompt templates are used to dynamically generate prompts tailored to the current stage of the orchestration process.
iii. API Client Initialization and Model Interaction:
    1. The system initializes API clients for multiple AI models, including Llama, ChatGPT, and Gemini.
    2. The initial prompt is sent to each configured AI model.
iv. Asynchronous Processing and Rate Limit Management:
    1. Asynchronous tasks manage API calls to external AI services.
    2. Rate limits are enforced to comply with API usage policies and prevent service interruption.Retry mechanisms with exponential backoff handle API request failures.
v. Initial Response Generation:Each AI model generates an initial response to the prompt.The system gathers these initial responses.
vi. Meta-Analysis:The initial responses are synthesized into a new prompt for meta-analysis.Each AI model analyzes the initial responses, aiming to extract insights or produce an improved version, resulting in meta-responses.
    1. A meta analysis step evaluates the evolution of ideas across the different layers of the process.
vii. Ultra-Synthesis:The meta-responses are further synthesized to form a more refined and insightful ultra-response.A designated "ultra engine" (selected AI model) is used for this synthesis.
viii. Hyper-Level Analysis:A hyper-level analysis is performed to synthesize insights from all previous stages into a final recommendation or analysis.
    1. This involves dissecting and refining the ultra-response.
ix. Output Formatting:Responses from AI models are formatted according to predefined templates.The output format can be customized (e.g., plain text, markdown, JSON).
x. Output and Storage:
    1. The synthesized output is saved for review or further processing.
xi. Outputs and system performance metrics are stored in a dedicated directory.

d. **Code Outline**: The code implementation of the UltrLLMOrchestrator revolves around a central class, UltrLLMOrchestrator, which orchestrates the interactions between multiple AI models. Key components and functionalities include:
i. Initialization:
    1. Setting up API clients for different AI services (Llama, ChatGPT, Gemini).
    2. Loading API keys from environment variables.
    3. Configuring rate limits for each AI model to comply with API usage policies.

4. Defining prompt templates for different stages of analysis (initial, meta, ultra, hyper).
5. Detecting and utilizing available hardware acceleration (e.g., Apple Silicon GPU).

ii. Orchestration Process:
1. Extracting keywords from prompts.
2. Generating initial responses from all configured AI models based on a user-defined prompt.
3. Performing meta-analysis of the initial responses to develop a deeper understanding and create a meta-response.
4. Synthesizing the meta-response to form a more refined and insightful ultra-response, utilizing a designated "ultra engine".
5. Conducting hyper-level analysis to cross-analyze and integrate insights from all preceding analytical stages, providing a comprehensive output.

iii. API Interaction and Management:
1. Asynchronous programming (asyncio) to manage API requests effectively.
2. Retry mechanisms with exponential backoff to handle API request failures gracefully.
3. Dynamic management of API rate limits to prevent overuse and ensure compliance with usage policies.

iv. Data Handling and Formatting:
1. Data classes (PromptTemplate, RateLimits) for managing prompt formats and API call frequencies.
2. Formatting responses according to predefined templates to facilitate structured synthesis across different analysis stages.
3. Saving responses and system performance metrics to a dedicated directory.

v. File Structure::
1. Main Orchestrator File: Contains the UltrLLMOrchestrator class and its methods for initializing API clients, managing prompts, orchestrating the AI models, and handling responses. This file would likely include the core logic for the entire process.

vi. Configuration Files:
1. .env file: Stores API keys and other sensitive information as environment variables.
2. Configuration files (potentially JSON or YAML): Store settings for prompt templates, rate limits, and other configurable parameters.
3. Data Classes File: Defines data structures such as PromptTemplate and RateLimits using the dataclass decorator. These classes manage different levels of analysis prompts and API rate limits.

vii. Modules for AI Model Interaction: Separate modules or functions for interacting with each AI model (Llama, ChatGPT, Gemini), encapsulating the API calls and response handling for each service.

viii. Utility Functions:
1. formatter: Formats the output based on the specified format (e.g., plain text).
2. _respect_rate_limit: Ensures that API rate limits are respected.
3. _setup_directory: Creates a directory for storing responses from each run.
4. Logging Module: Configures logging to track system operations, errors, and performance metrics.
5. Hardware Detection Module: Detects available hardware accelerations (e.g., Apple's Metal for MPS backends) to enhance performance.

ix. Response Storage: A directory (responses) to store the generated outputs from each stage of the orchestration process. Each run may have its own subdirectory based on the prompt.
1. This structure supports modularity, making it easier to maintain, extend, and adapt the system to new AI models or changing requirements.

e. Code Samples:[The sources contain code snippets, but not a full implementation. Insert code samples illustrating key aspects of the invention, such as API initialization, prompt orchestration, and response synthesis. This would involve Python code utilizing libraries like asyncio, requests, tenacity, and dotenv.]

f. Brief Description of the Drawwings: The following references the several drawings and diagrams that are offered in support of this provisional patent aopplication

08. Potential variations and applications of the UltrLLMOrchestrator system that could be considered:
a. arying AI Model Selection The system can dynamically select AI models based on configured priorities or task requirements1.... This could involve using different combinations of

models (e.g., only Llama and ChatGPT, or including other AI models beyond the mentioned three) or prioritizing specific models for certain types of tasks[2]....

b. Customizable Processing Pipelines The system can be modified to accommodate the unique requirements of each model[6].... This might involve adjusting input/output formats, integrating different processing pipelines, or modifying the orchestration logic to better suit the capabilities of specific AI models[6].
c. Alternative Orchestration Workflows The multi-layered processing workflow can be altered[5].... Rather than strictly adhering to the initial, meta, ultra, and hyper-level analysis sequence, the system could support different sequences, parallel processing of certain stages, or conditional branching based on the content of the responses[8]....
d. Hardware Optimization Customization The system is capable of detecting and leveraging hardware acceleration, such as Apple Silicon GPUs[3].... The system could be adapted to support other forms of hardware acceleration or to allow users to manually configure hardware settings to optimize performance for their specific environment[14]....
e. Output Format Flexibility The system can format responses from AI models according to a configurable output format[2].... Different output formats like plain text, markdown, or JSON can be supported to cater to different downstream applications and user preferences[2].... The system could be extended to support other output formats or to allow users to define their own custom formats[2]....
f. Varying the Orchestrator Engine The orchestrator may use multiple engines, such as Llama, ChatGPT, or Gemini, which offer unique capabilities and advantages[27].... By combining these engines, the invention creates a hybrid system that optimizes response quality and diversity[27]
g. These variations highlight the flexibility and adaptability of the UltrLLMOrchestrator system, allowing it to be tailored to different use cases, hardware configurations, and user preferences[30]....
h. New UX UI can focus on different hting s
i. Orchestrator thechnology as inteligence multiplier as applied to speciic scenarios
j. The manner in which the models are exposed to the work of the other models and the instructions for how to comple each level draft
k. The inclusion of an expert compiling model to create the last Ultra level result by combining the
l. Orchestrator thechnology as inteligence multiplier as applied to speciic scenarios as AGI over singular model
m. AGI regulator
n. Machine learning enhancement -- multiplying the self adjusting parameters to increase efficiency
o. Deep Neural enhancement augmentation
p. Program processing speed multiplier

The sources suggest the UltrLLMOrchestrator system could be a step toward Artificial General Intelligence (AGI) by synergistically combining the strengths of individual AI models, although they do not explicitly claim that the system achieves AGI. Here's how the system's architecture and capabilities align with AGI concepts:

- Comprehensive and Multi-faceted Analysis: By orchestrating multiple AI models such as Llama, ChatGPT, and Gemini, the system can perform a more comprehensive and nuanced analysis of complex problems than any single model could achieve. Each model contributes its unique strengths, leading to a multi-layered analysis that incorporates diverse perspectives.

- Emergent Synthesis: The system's multi-stage process—initial response, meta-analysis, ultra-synthesis, and hyper-analysis—allows for the progressive refinement of insights, potentially uncovering emergent patterns and novel perspectives that wouldn't be accessible to individual models. This synthesis of insights from different models reflects a higher-order understanding, moving beyond the limitations of individual AI capabilities.

- Dynamic Adaptability: The UltrLLMOrchestrator dynamically adapts to various factors, such as:

  - API rate limits.
  - Hardware configurations.
  - Specific task requirements.
- This adaptability enables the system to function efficiently across different environments and handle a wide range of tasks.

- Meta-Learning and Self-Improvement: While not explicitly stated, the system's architecture could facilitate meta-learning, where the system learns from the responses of individual models and improves its performance over time. The integration of a feedback loop, where AI models learn from the analysis of their own responses, further enhances this capability. The system’s ability to perform self-analysis for patent purposes demonstrates a reflective capability and self-awareness.

- Mimicking Cognitive Processes: The tiered approach to analysis attempts to mimic human cognitive processes by breaking down complex problems into smaller, manageable tasks and synthesizing information from multiple sources to arrive at a comprehensive understanding.

By combining these elements, the UltrLLMOrchestrator represents a step towards AGI by creating a system that integrates and orchestrates multiple specialized AI models to achieve a more general and comprehensive intelligence. The system's ability to dynamically adapt, learn, and synthesize information from diverse sources reflects key aspects of general intelligence.

1.

q. Novel Aspects and Advantages over prior efforts :

i. Synthesized Analysis: Orchestrates multiple AI models to produce a synthesized analysis of a query or prompt.
ii. Dynamic Rate Limit Management: Manages rate limits dynamically for each AI model.
iii. Emergent Pattern Identification: Cross-analyzes and integrates insights from all preceding analytical stages to identify emergent patterns and novel perspectives.
iv. Comprehensive Output: Provides a comprehensive output that captures emergent patterns and novel perspectives unique to the combined AI analysis..
v. Comprehensive Multi-Model Orchestration The system integrates multiple AI models (Llama, ChatGPT, and Gemini) to leverage their collective strengths, generating more accurate and comprehensive responses than individual AI models can produce.
vi. Novelty and Non-Obviousness The system orchestrates multiple AI models to perform a comprehensive analysis of a given prompt, leveraging the unique strengths of different models to produce a multi-layered analysis.
vii. Multi-Layered Analysis and Synthesis The system employs a multi-layered processing workflow that includes initial response generation, meta-analysis, ultra-synthesis, and hyper-level analysis to refine and synthesize responses, leading to a more comprehensive and insightful output. This goes beyond systems that rely on singular model outputs or manual integration processes.
viii. Dynamic Rate Limit Management The system incorporates dynamic rate limit management for each AI model, ensuring compliance with external API usage policies. It manages and respects the rate limits of different AI models/APIs, mitigating the risk of service interruption due to excessive requests.
ix. Hardware Optimization The system automatically detects and utilizes available hardware acceleration (e.g., Apple Silicon's MPS) to optimize model inference performance, adapting to diverse deployment environments.
x. Flexible and Configurable Design The system's responses from AI models are formatted according to a configurable output format. The architecture supports a flexible design that can be adapted to include additional models as required.
xi. Self-Analysis and Documentation Generation The orchestration process includes a proprietary step to analyze and save its operational metadata and results for patent analysis. This demonstrates the system's capability to participate in self-analysis and documentation generation.
xii. Efficient Resource Utilization The system utilizes asynchronous processing techniques for parallel execution of API calls and orchestration tasks, significantly improving the efficiency of the orchestration process.
xiii. Robust Error Handling Incorporation of retry functionality with exponential backoff handles occasional API request failures gracefully, enhancing the system's resilience and reliability.
xiv. Novel Code Implementation The invention further comprises a novel code implementation that enables efficient and seamless integration of these models. The code streamlines the development and deployment of language models, making it easier for developers to create more sophisticated and accurate responses.

09. Claims include:

- A method for orchestrating multiple artificial intelligence models to perform multi-tier analyses and synthesize responses from initial, meta, ultra, to hyper-level perspectives.
- The system of claim i, wherein the orchestration includes rate limit management, dynamic hardware detection, and asynchronous task execution.
- The system of claim i, wherein the orchestration uses custom templates for generating structured prompts to guide the multi-tier analysis process.
- The system of claim i, wherein the orchestration includes a mechanism for dynamically selecting an AI model based on task requirements for final synthesis.

10. Conclusion:

The UltrLLMOrchestrator system represents a significant advancement in AI-driven text analysis and synthesis. Its innovative approach to integrating responses from multiple AI models, coupled with enhanced processing capabilities and efficient management of interactions, provides a versatile and powerful tool for generating synthesized and nuanced interpretations of input data. This system opens new avenues for exploring multi-AI collaborations, benefiting various applications in research, content generation, analysis, and beyond.

This provisional patent application outlines the essential elements of the UltrLLMOrchestrator system, establishing priority for the described invention. Further refinement and formalization will be pursued in a subsequent non-provisional patent application.

11. Glossary of terms

Here is a glossary of terms, as applied in the context of the provided patent excerpts, that may be ambiguous, novel, or used differently in other contexts:

- **AI Orchestration System**: A system designed to manage and integrate multiple AI models to perform tasks in a coordinated manner. This involves managing workflows, data processing, and synthesis of outputs from different AI models.
- **Generative AI Models**: AI models, such as Google's Gemini and OpenAI's GPT, that are capable of generating new content, including text, images, and other media.
- **Initial, Meta, Ultra, and Hyper-Level Analyses**: A multi-stage process of analyzing and synthesizing responses from AI models:
    - **Initial Analysis**: The first-level responses generated by AI models to a given prompt.
    - **Meta-Analysis**: The analysis of initial responses to improve and refine them.
    - **Ultra-Analysis**: The synthesis of meta-analyses to create a single, holistic response.
    - **Hyper-Analysis**: A higher-order synthesis of all previous responses, providing a comprehensive and integrated output.
- **TriLLMOrchestrator**: The name of the AI model orchestration system that is the subject of the patent application. It is designed to manage the workflow between multiple AI generative models to perform analysis, meta-analysis, and synthesis of data or textual prompts in an automated, cohesive process.
- **Rate Limits**: Restrictions on the number of requests that can be made to an API within a certain time period. The system manages these limits to ensure compliance with API usage policies.
- **Hardware Acceleration**: The use of specialized hardware, such as GPUs (Graphics Processing Units), to speed up the processing of AI models. The system detects available hardware resources and utilizes them to optimize performance.
- **Prompt Templates**: Configurable templates used to format prompts for AI models. These templates are tailored to elicit specific types of analysis from the models.
- **Self-Analysis**: The system's capability to analyze its own functionality for documentation and patent application purposes.
- **API Clients**: Software components that enable communication with AI services. These clients handle the initialization, request submission, and response retrieval from AI models.
- **Orchestrator**: A component that integrates multiple language models to create more accurate and informative responses.
- # Technical Glossary for Patent Documentation
-
- **Context**: Comprehensive information associated with user inputs, prompts, responses, and interactions within the system, including but not limited to: conversation history, user session data, previous analyses, schema of analyzed data, user roles, and system-specific contextual parameters. Context may be stored and managed by a dedicated context module for use in service selection and request formatting.
-
- **Data Object**: A structured data container representing specific real-world entities, events, or documents, comprising: (1) a unique identifier, (2) definable properties and attributes, (3) metadata including geographic locations, associated values, probabilities, and events, and (4) relationships to other data objects within the system architecture.
-
- **Data Processing Service**: A computational system component that: (1) receives and responds to data processing requests via exposed APIs, (2) implements specific functionalities including but not limited to search, filtering, indexing, formatting, and database operations, and (3) communicates with Service Orchestrators through standardized protocols.
-
- **Language Model**: An algorithmic system that: (1) predicts probability distributions of word sequences, (2) generates contextually relevant text based on training patterns, (3) processes natural language inputs, and (4) implements various architectures including n-gram, exponential, positional, or neural network models.
-
- **Large Language Model (LLM)**: An advanced language model implementation characterized by: (1) extensive training parameters, (2) capability to process multiple modalities, (3) transformer-based architecture with attention mechanisms, (4) self-supervised and/or semi-supervised learning

frameworks, and (5) integration capabilities with security and permission systems. Examples include BERT, GPT series, PaLM, and LLAMA.

- **Model**: A computer-implemented system comprising: (1) sequential, functional, or concurrent processing capabilities, (2) computational frameworks including neural networks, language models, and artificial intelligence systems, (3) multimodal input processing capabilities, and (4) defined input-output relationships.
- **Ontology**: A structured data representation system that defines: (1) data object types and associated property types, (2) link types and relationships between data objects, (3) action definitions and property value modifications, (4) hierarchical relationships, and (5) executable functions associated with specific data object types.
- **Orchestration System**: An integrated framework that: (1) manages component interactions, (2) implements Orchestrator Selector functionality, (3) coordinates multiple Service Orchestrators, and (4) facilitates communication between various system services through standardized protocols.
- **Prompt**: A natural language input mechanism comprising: (1) phrases, questions, or statements in human language, (2) user-generated or automatically generated content, (3) task specifications and data requests, and (4) optional tool and data object type indicators.
- **Service Orchestrator**: A system component that: (1) generates formatted service requests, (2) implements service-specific logic and rules, (3) processes natural language tasks into structured queries, and (4) manages service specifications including input parameters, formats, and response handling.

12. Visual Appendicies

-